VertEgg – A toolbox for simulation of vertical distributions of fish eggs
Version 1.0

Bjørn Ådlandsvik
Institute of Marine Research
P.O. Box 1870, Nordnes
N-5024 Bergen
Norway
E-mail: bjorn@imr.no

January 26, 2000

Contents

Introduction

The vertical distribution of fish eggs in a water column is determined by the buoyant forcing and turbulent mixing. A model of the steady state distribution has been developed by Sundby (1983). Westgård (1989) developed two numerical models for the time evolution of the concentration. These models are written in FORTRAN and use the GPGS library for graphics.

The commercial software system Matlab and the free alternative Octave provide interactive command-line environments for numerical computations and visualisation. VertEgg is a toolbox for scientific work on vertical egg distributions in these environments. It contains tools for analysing observed distributions and performing numerical simulations including pre- and post-processing of the results. VertEgg is *not* a finished application, the tools are intended to be used interactively or sewed together by the user to make the application that solves her problem. This approach should give the competent user a powerful and flexible environment for research on vertical distributions.

The author believes strongly in learning by problem solving and examples. Therefore the main part of the manual is the example chapter. The examples ranges from simple illustrations of the functions in the toolbox, by verification and sensitivity studies to a ready-to-run egg-distribution model complete with file I/O. These examples are also available online as *scripts*, programs in the internal programming language of the packages. The recommended way to make small applications is to take the closest example script and try out modifications.

The first two chapters give an introduction to the theory of vertical egg distributions and the numerical method used. The manual also has a chapter containing the reference manual, a systematic description of the tools provided in VertEgg.

Chapter 1

Theory of Vertical Egg Distributions

1.1 The Equations

Neglecting horizontal processes, the evolution of the vertical distribution of a class of fish eggs is governed by a simple *conservation* principle

$$\begin{aligned} &\text{The change in the number of eggs between two depth levels} \\ &\quad = \text{the number of eggs entering at the lower level} \\ &\quad - \text{the number of eggs leaving at the upper level} \\ &\quad + \text{the number of eggs spawned} \\ &\quad - \text{the number of eggs that hatched or died} \end{aligned} \tag{1.1}$$

To put this in a more mathematical formulation, let the variable z denote depth and t time. The z-axis is chosen to point *up*, that is $z = 0$ at the surface and negative values are used in the water column. This choice is opposite from Sundby (1983, 1991) and Westgård (1989). The present choice is motivated by the most common axis-convention in 3D hydrodynamic ocean modelling.

To make the theory more convenient mathematically, a *continuum* approach is used, where the eggs are replaced by a continuous egg *concentration*. Let $\varphi = \varphi(z,t)$ denote this egg concentration at depth z and time t. Let $F = F(z,t)$ be the upwards egg *flux*, i.e. the net number of eggs passing upwards per time unit at depth z and time t. Similarly, let $Q = Q(z,t)$ denote the *source term*, the number of eggs spawned minus the numbers that hatched or died per depth and time unit at depth z and time t. The source term can also be used to take care of other processes, such as loss of eggs by horizontal advection.

The conservation principle 1.1 can then be formulated in *integral form*

$$\int_{z_1}^{z_2} \varphi(z,t_2)\,dz - \int_{z_1}^{z_2} \varphi(z,t_1)\,dz = \int_{t_1}^{t_2} F(z_1,t)\,dt - \int_{t_1}^{t_2} F(z_2,t)\,dt + \int_{t_1}^{t_2}\int_{z_1}^{z_2} Q(z,t)\,dzdt \tag{1.2}$$

Mathematically this can be reformulated in mixed form, differential in time and integral in space

$$\frac{\partial}{\partial t}\left(\int_{z_1}^{z_2}\varphi(z,t)\,dz\right) = F(z_1,t) - F(z_2,t) + \int_{z_1}^{z_2} Q(z,t)\,dz \quad \text{for } -H \le z_1 < z_2 \le 0. \tag{1.3}$$

or in pure *differential* form

$$\frac{\partial \varphi}{\partial t} = -\frac{\partial F}{\partial z} + Q \tag{1.4}$$

The flux is decomposed in two parts, the *convective flux* and the *diffusive flux*. The convection is due to the terminal buoyant velocity $w = w(z,t)$ of the egg which is computed by the density difference from the surroundings and the egg size as described in section 1.3. The formulation is simply,

$$F^{\text{conv}} = w\varphi. \tag{1.5}$$

The diffusion is caused by turbulent mixing and is modelled by Fick's law using the vertical eddy diffusion $K = K(z,t)$,

$$F^{\text{diff}} = -K\frac{\partial \varphi}{\partial z}. \tag{1.6}$$

Often the source term Q can be separated in a *spawning* or *production* term independent of the egg concentration

$$Q^{\text{spawn}} = P \tag{1.7}$$

and a *mortality* or *loss* term depending on the concentration

$$Q^{\text{loss}} = -\alpha\varphi \tag{1.8}$$

Using this the differential conservation law (1.4) becomes

$$\frac{\partial \varphi}{\partial t} = -\frac{\partial}{\partial z}(w\varphi) + \frac{\partial}{\partial z}(K\frac{\partial \varphi}{\partial z}) + P - \alpha\varphi \tag{1.9}$$

This parabolic partial differential equations is called the *convection-diffusion* or *transport* equation.

The boundary conditions are simply no flux across the surface and bottom,

$$F(0,t) = F(-H,t) = 0, \tag{1.10}$$

and the initial condition is given by

$$\varphi(z,0) = \varphi_0(z), \quad -H \le z \le 0. \tag{1.11}$$

1.2 Solutions of the equations

Under simplified circumstances the equations in section 1.1 can be solved analytically. This section examines some of these solutions.

1.2.1 Vertical integrated equation

Let Φ denote the total concentration, $\Phi = \int_{-H}^{0} \varphi dz$. Take the vertical integral of equation (1.9) and use the boundary conditions (1.10) gives

$$\frac{d\Phi}{dt} = P^{tot} - \int_{-H}^{0} \alpha\varphi \, dz \tag{1.12}$$

where $P^{tot} = \int_{-H}^{0} P dz$ is the total spawning contribution. Without any source terms the total concentration Φ is constant. To reach a steady state solution with spawning, the loss term α must be nonzero.

If α is a positive constant, the solution to the equation above is

$$\Phi(t) = \mathrm{e}^{-\alpha t}\Phi(0) + (1 - \mathrm{e}^{-\alpha t})\frac{P^{tot}}{\alpha} \tag{1.13}$$

with steady state solution $\Phi = P^{tot}/\alpha$. If $\alpha = 0$ the solution is simply

$$\Phi(t) = \Phi(0) + tP^{tot}. \tag{1.14}$$

1.2.2 Stationary solution

A stationary solution solves the conservation law (1.4) without the time derivative

$$\frac{dF}{dz} = Q \tag{1.15}$$

with boundary conditions $F(0) = F(-H) = 0$ given by (1.10). The condition for the existence of this solution is that the total source term vanishes,

$$Q^{tot} = \int_{-H}^{0} Q \, dz = 0 \tag{1.16}$$

the flux function is then given by integrating (1.15)

$$F(z) = \int_{-H}^{z} Q(s) \, ds = -\int_{z}^{0} Q(s) \, ds. \tag{1.17}$$

Without source term the expression above reduces to $F = 0$, i.e. the net flux is zero everywhere. This simply says that in a steady state convection is balanced by diffusion

$$w\varphi - K\frac{\partial\varphi}{\partial z} = 0. \tag{1.18}$$

This is an ordinary differential equation for $\varphi(z)$. The boundary conditions (1.10) does not contain more information. A unique solution can nevertheless be singled out by the integral condition

$$\int_{-H}^{0} \varphi(z)\, dz = \Phi. \tag{1.19}$$

In other words, among all the solutions of (1.18) choose the one with correct vertically integrated concentration.

To simplify the notation put $m = w/K$ and let $M(z) = -\int_z^0 m(s)ds$. Then the stationary solution is

$$\varphi(z) = \frac{\Phi}{\int_{-H}^{0} \mathrm{e}^{M(s)} ds} \mathrm{e}^{M(z)}. \tag{1.20}$$

Constant coefficients

The case with constant coefficients was studied by Sundby (1983). In this case $M(z) = mz$ and the solution is a truncated exponential distribution,

$$\varphi_m(z) = \Phi \frac{m}{1 - \mathrm{e}^{-mH}} \mathrm{e}^{mz}. \tag{1.21}$$

In the toolbox, this solution is computed by the function `eggsact`. This solution has the following symmetry between ascending and descending velocity,

$$\varphi_{-m}(z) = \varphi_m(-H - z) \tag{1.22}$$

With large depth and/or high ascending velocity the effect of the bottom may be neglected. In this case the distribution is well approximated by an exponential distribution with parameter $\lambda = 1/m$, that is

$$\varphi_m(z) \approx \Phi m \mathrm{e}^{mz}, \quad \text{for } mH \gg 1. \tag{1.23}$$

For positive values of m the following series development from (Sundby, 1983) is valid

$$\varphi_m(z) = \Phi m \mathrm{e}^{mz} \sum_{j=0}^{\infty} \mathrm{e}^{-jmH}, \quad m > 0. \tag{1.24}$$

The zero-th term of this expansion is the exponential distribution above. For negative values of m the symmetry relation (1.22) can be used,

$$\varphi_m(z) = -\Phi m \mathrm{e}^{mz} \sum_{j=1}^{\infty} \mathrm{e}^{jmH}, \quad m < 0. \tag{1.25}$$

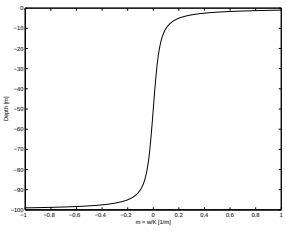

Figure 1.1: Mean depth μ as a function of m

To compute the mean and variance of the distribution the constant factor Φ can be dropped. The moment generating function is then

$$\psi(t) = \int_{-H}^{0} \frac{m}{1 - \mathrm{e}^{-mH}} \mathrm{e}^{(m+t)z} dz = \frac{m}{1 - \mathrm{e}^{-mH}} \frac{1 - \mathrm{e}^{-(m+t)H}}{m + t}. \tag{1.26}$$

Differentiating ψ gives the moments of the distribution. The mean depth is given by

$$\mu = \psi'(0) = -\frac{1}{m} + \frac{H}{\mathrm{e}^{mH} - 1} \tag{1.27}$$

The function $\mu = \mu(m)$ with $H = 100\,\mathrm{m}$ is plotted in figure 1.1. The variance is

$$\psi''(0) - \mu^2 = \frac{2 - \mathrm{e}^{-mH}(m^2H^2 + 2mH + 2)}{m^2(1 - \mathrm{e}^{-mH})} - \mu^2 \tag{1.28}$$

Piecewise constant coefficients

The solution (1.21) can easily be extended to a piecewise constant m. Let $0 = z_0 > \cdots > z_m = -H$ be a partition of the water column with $m(z) = m_i$ on the interval $I_i = (z_i, z_{i-1})$. Then the stationary solution restricted to the interval I_i is given as

$$\varphi(z) = C_i \mathrm{e}^{m_i z}, \quad \text{for } z_i < z < z_{i-1} \tag{1.29}$$

The C_i-s are determined by continuity of the solution

$$C_i \mathrm{e}^{m_i z_i} = C_{i+1} \mathrm{e}^{m_{i+1} z_i}, \quad i = 1, \ldots, m-1 \tag{1.30}$$

and the integral condition (1.19)

$$\sum_{i=1}^{m} C_i \int_{z_i}^{z_{i-1}} \mathrm{e}^{m_i z} dz = \sum_{i=1}^{m} C_i \frac{\mathrm{e}^{m_i z_{i-1}} - \mathrm{e}^{m_i z_i}}{m_i} = \Phi. \tag{1.31}$$

This method is used in the function `sstate` to approximate the general steady state solution. More details of the implementation is given in section 2.2.

Linear coefficients

In Sundby (1991) the solution is derived for m linear. Let m be given as $m(z) = a(z - z_0)$. Then $M(z) = \frac{1}{2}a((z - z_0)^2 - z_0^2)$. Taking the last term into the constant C, the solution becomes

$$\varphi(z) = C\mathrm{e}^{\frac{1}{2}a(z-z_0)^2}. \tag{1.32}$$

Bathypelagic eggs are neutral buoyant at $z = z_0$, raising if deeper and sinking if higher in the water column. In this case the coefficient a is negative and the concentration has a normal distribution about $z = z_0$ with variance $\sigma^2 = 1/|a|$.

Stationary solution with source terms

With source terms, the steady state equation is

$$0 = -\frac{\partial}{\partial z}(w\varphi) + \frac{\partial}{\partial z}(K\frac{\partial \varphi}{\partial z}) + P - \alpha\varphi. \tag{1.33}$$

This is a general second order ordinary differential equation. With constant coefficients the equation becomes

$$K\varphi'' - w\varphi' - \alpha\varphi = -P \tag{1.34}$$

and the no flux boundary conditions are

$$K\varphi' - w\varphi, \quad z = 0, z = -H. \tag{1.35}$$

The solution can be written

$$\varphi(z) = A\mathrm{e}^{az} - B\mathrm{e}^{-bz} + \frac{P}{\alpha} \tag{1.36}$$

with

$$a = \frac{1}{2K}(\sqrt{w^2 + 4\alpha K} + w) \tag{1.37}$$

$$b = \frac{1}{2K}(\sqrt{w^2 + 4\alpha K} - w) \tag{1.38}$$

$$A = \frac{wP}{\alpha bK}\frac{1 - \mathrm{e}^{-bH}}{1 - \mathrm{e}^{-(a+b)H}} \tag{1.39}$$

$$B = \frac{wP}{\alpha aK}\mathrm{e}^{-bH}\frac{1 - \mathrm{e}^{-aH}}{1 - \mathrm{e}^{-(a+b)H}}. \tag{1.40}$$

This function is computed in the toolbox by the function `srcsact`. The signs are chosen such that positive velocity w makes all terms positive. For negative velocities a symmetry property similar to equation (1.22) can be used, if $\varphi(z)$ is a solution to equation (1.34) and boundary conditions (1.10) then $\varphi(-H-z)$ is a solution the same equation and boundary conditions with the opposite sign on w. In other words,

$$\varphi_{-w}(z) = \varphi_w(-H-z). \tag{1.41}$$

1.3 The Terminal Egg Velocity

An egg in sea water will reach its terminal velocity w, where the buoyant forcing balances the frictional drag. This velocity is a function of the difference $\Delta\rho = \rho - \rho_e$ between the density of the water and the egg, the egg diameter d, the acceleration g due to gravity and the molecular viscosity. Here the *dynamic* molecular viscosity μ is used.

The situation is characterised by the non-dimensional *Reynolds number*,

$$Re = \frac{\rho d w}{\mu} \tag{1.42}$$

For low values, $Re < 0.5$, the terminal velocity is given by *Stokes' formula*

$$w = \frac{1}{18}\frac{g d^2 \Delta\rho}{\mu}. \tag{1.43}$$

This formula was obtained by Stokes (1851). The derivation of the formula is given in almost any textbook on fluid dynamics, for instance (Yih, 1977).

Combining these equations, one obtains an expression for D, the maximum diameter for which Stokes' velocity applies,

$$D^3 = \frac{9\mu^2}{\rho g \Delta\rho} \tag{1.44}$$

In the intermediate region $0.5 < Re < 5$, Dallavalle (1948) gave an empirical formula

$$w = K_I(d - \zeta D)\Delta\rho^{2/3}\mu^{-1/3} \tag{1.45}$$

where $\zeta = 0.4$ for a sphere. The coefficient K_I is determined by the requirement that both formulas should give the same answer for $Re = 0.5$ or equivalently $d = D$. This gives

$$K_I = \frac{5}{54} 9^{1/3} g^{2/3} \rho^{-1/3} = 0.0875 \mathrm{kg}^{-1/3}\,\mathrm{m}^{5/3}\mathrm{s}^{-4/3}. \tag{1.46}$$

The formulas above are implemented as the function `eggvel` in the VertEgg toolbox. The buoyancy of a fish egg is often given as the salinity S_e where the egg is neutrally buoyant. The function `eggvelst` computes the terminal velocity in this case.

To compute the egg velocity the density, ρ of water is needed. In the toolbox only density at surface pressure is presently available. This is computed by the function `dens0` by the UNESCO formula (UNESCO, 1981). The function `sw_dens` in the SEAWATER toolbox (Morgan, 1994) has implemented the full UNESCO equation of state.

The dynamic molecular viscosity μ of sea water is tabulated in table 1.1 taken from Sverdrup *et al.* (1952). The values decrease with temperature and increase slowly with salinity. The dependence on pressure is insignificant, and is neglected here.

Salinity	Temperature [°C]						
[psu]	0	5	10	15	20	25	30
0	1.79	1.52	1.31	1.14	1.01	0.89	0.80
10	1.82	1.55	1.34	1.17	1.03	0.91	0.82
20	1.85	1.58	1.36	1.19	1.05	0.93	0.84
30	1.88	1.60	1.38	1.21	1.07	0.95	0.86
35	1.89	1.61	1.39	1.22	1.09	0.96	0.87

Table 1.1: Dynamic molecular viscosity of sea water with unit $10^{-3}\,\mathrm{kgm^{-1}s^{-1}}$

Using linear least squares regression the table is approximated by the following function where T is the temperature in °C and S is the salinity in psu.

$$\mu = 10^{-3}\,(1.7915 - 0.0538\,T + 0.007\,T^2 - 0.0023\,S)\,\mathrm{kgm^{-1}s^{-1}}, \tag{1.47}$$

This reproduces table 1.1 with an absolute error less than $2 \times 10^{-5}\,\mathrm{kgm^{-1}s^{-1}}$ and a relative error of 1.7 %. This is good enough to compute egg velocities where the uncertainty in the other variables are larger. This formula is implemented by the function `molvisc` in the toolbox. Riley and Skirrow (1975) give a more precise formula which requires more computational effort.

Chapter 2

Numerical Methods

The numerical solutions are computed on a fixed equidistant grid. The number of grid cells is denoted N and the cell height is Δz. The grid is staggered as shown in fig. 2.1. The z-axis points upwards.

The cell interfaces, the *flux points*, are

$$z_i = (1 - i)\Delta z \quad \text{for} \quad i = 1, \ldots, N + 1. \tag{2.1}$$

The cell centers, the *concentration points* or the *egg points*, are

$$\bar{z}_i = \frac{z_i + z_{i+1}}{2} = (\frac{1}{2} - i)\Delta z \quad \text{for} \quad i = 1, \ldots, N. \tag{2.2}$$

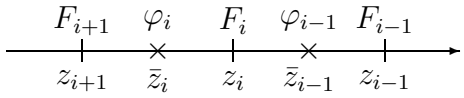

Figure 2.1: Horizontal view of the vertical grid

The variables are discretized in the following ways. The egg concentration φ_i represents the mean concentration in cell i, that is

$$\varphi_i \Delta z = \int_{z_{i+1}}^{z_i} \varphi(z)\, dz, \quad i = 1, \ldots, N. \tag{2.3}$$

The φ_i values may be considered living on the egg points $\bar{z}_i$ as in fig. 2.1. As the flux values are computed from the vertical velocity and the eddy diffusivity, F_i, w_i and K_i are taken as point values on the flux points z_i for $i = 1, \ldots, N + 1$. The source terms P_i and α_i are cell averages and live on the egg points.

2.1 The Transient Problem

The numerical solution of the convection-diffusion equation (1.4) are covered by several authors. A classic source is Roache (1972). A newer source is chapter 9 in the book by Fletcher (1991). Recently a whole book, edited by Vreugdenhil and Koren (1993), has been devoted to this equation. For conservative methods, as will be used here, the book by LeVeque (1992) is also recommended.

For our problem, convection and diffusion of concentration of fish eggs, some properties of the numerical method is important. Fish eggs are usually found in a subrange of the water column with values close to zero outside this range. And of course a real concentration do not have negative values. The method must therefore be *positive*, that is it must not create any negative values.

In the absence of source and sink terms, eggs are not created or destroyed. The vertical integrated concentration is therefore constant in time as shown in section 1.2.1. The numerical method should have the same property, it should be *mass conserving*.

Of course the numerical solution should be as close as possible to the real solution. That is the *accuracy* of the method should be good. This concept includes low artificial diffusion and no or very limited wiggles development.

A convection – diffusion problem is characterised by the following non-dimensional numbers, all defined at the interior flux points, z_i for $i = 2, \ldots, N$.

$$\text{Courant number} \qquad c_i = w_i \frac{\Delta t}{\Delta z} \tag{2.4}$$

$$\text{Diffusive parameter} \qquad s_i = K_i \frac{\Delta t}{\Delta z^2} \tag{2.5}$$

$$\text{Cell Peclet number} \qquad P_i^{\text{cell}} = \frac{|w_i|\Delta z}{K} = \frac{|c_i|}{s_i} \tag{2.6}$$

If the coefficients w and K are constant, the subscripts are simply dropped.

For fish eggs, the terminal velocity is in the order of a couple of millimetres per second. The velocity can be positive (*pelagic* eggs), negative (*benthic*) or change sign some place in the water column (*mesopelagic*). The vertical eddy diffusion show a very large range of variation, from more than 10^{-2} $\mathrm{m^2 s^{-1}}$ in the upper mixing layer to $10^{-5}\,\mathrm{m^2 s^{-1}}$ in the pycnocline layer. With a grid size of the order of one meter, the Cell Peclet number P^{cell} take values from 10^{-1} to 10^2. This means that the process may be dominated by diffusion in the upper layer and convection below.

The numerical methods considered here are *finite difference* or really *finite volume* schemes given in *conservation form* (or *flux formulation*). These methods are conceptually natural, working directly with the integral form of the conservation law (1.2). More precisely, the equation (1.2) is approximated by

$$(\varphi_i^{n+1} - \varphi_i^n)\Delta z = (F_{i+1}^n - F_i^n)\Delta t + Q_i^n \Delta z \Delta t, \quad i = 2, \ldots, N \tag{2.7}$$

where the superscripts indicate the time step. The boundary conditions (1.10) simply become $F_1 = F_{N+1} = 0$. The various methods differ in their estimation of the total flux

function $F_i = F_i^{\text{conv}} + F_i^{\text{diff}}$ and the source term Q_i. One advantage of this flux formulation is that mass conservation is automatically fulfilled, as the fluxes cancel out during vertical integration.

Without source term the total concentration is conserved. Therefore the local concentration values can not become arbitrary large, unless there are negative values in some other cells. In other words, positivity is a sufficient (but not necessary) condition for stability of such schemes.

In the following several methods are described. They are implemented in the toolbox as `ftcs`, `lwendrof`, `upstream`, `posmet`, and `minlim` respectively. Their performance will be studied in the example section 4.3. Several methods for the same problem causes a new problem, which method to choose. This depends on the range of P^{cell} values in the problem. A general advice is to try Lax-Wendroff first. This is an accurate and fast method when applicable. If oscillations occur, go for the flux-limited methods or use higher spatial resolution. In both cases more computing time is required.

2.1.1 Linear conservative schemes

The linear conservative schemes considered here estimates the total flux function in the form

$$F_i = \alpha_i \varphi_i + \beta_i \varphi_{i-1}, \tag{2.8}$$

where the α_i-s and β_i-s are independent of the φ_i-s. Sometimes this will be used in non-dimensional form

$$F_i = \frac{\Delta z}{\Delta t}(a_i \varphi_i + b_i \varphi_{i-1}). \tag{2.9}$$

Some general theory for this kind of numerical schemes is presented in Appendix B.2

The FTCS scheme

The simplest scheme is often called FTCS (forward time central space) after the differencing. This scheme is described for instance in chapter 9.4 in Fletcher (1991). Here both flux components are centred around $z = z_i$,

$$F_i^{\text{conv}} = w_i \frac{\varphi_{i-1} + \varphi_i}{2} \tag{2.10}$$

$$F_i^{\text{diff}} = -K_i \frac{\varphi_{i-1} - \varphi_i}{\Delta z}. \tag{2.11}$$

In the notation of equation (2.8) the method is given by

$$\alpha_i = \frac{1}{2} w_i + \frac{K_i}{\Delta z}, \quad \beta_i = \frac{1}{2} w_i - \frac{K_i}{\Delta z}, \tag{2.12}$$

and non-dimensionalised by

$$a_i = \frac{1}{2}c_i + s_i, \quad b_i = \frac{1}{2}c_i - s_i. \tag{2.13}$$

The following positivity condition for the FTCS scheme follows from the general condition in Appendix B.2.2.

$$|c_i| \leq 2s_i, \quad i = 2, \ldots, N \tag{2.14}$$

$$s_2 \leq 1 + \frac{1}{2}c_2 \tag{2.15}$$

$$s_i + s_{i+1} \leq 1 + \frac{1}{2}(c_i - c_{i+1}), \quad i = 2, \ldots, N-1 \tag{2.16}$$

$$s_N \leq 1 - \frac{1}{2}c_N \tag{2.17}$$

The first condition can be restated as $P^{\text{cell}} \leq 2$.

With constant coefficients, the positivity conditions reduce to

$$|c| \leq 2s \leq 1, \tag{2.18}$$

which is stronger than the stability condition

$$c^2 \leq 2s \leq 1. \tag{2.19}$$

The numerical "diffusion" is negative,

$$K_{num} = -\frac{1}{2}w^2\Delta t \tag{2.20}$$

which is negligible if $c^2 \ll 2s$.

This method is implemented in the toolbox as the function `ftcs`.

The Lax-Wendroff Scheme

An alternative is the Lax-Wendroff scheme as analysed for instance in Vreugdenhil (1993). The scheme compensates for the negative numerical "diffusion" in FTCS. The diffusive flux is still given by equation (2.11) but the convective flux is modified

$$F_i^{\text{conv}} = w_i\frac{\varphi_{i-1} + \varphi_i}{2} - \frac{1}{2}w_i^2\frac{\Delta t}{\Delta z}(\varphi_{i-1} - \varphi_i). \tag{2.21}$$

In the notation of equation (2.8) the Lax-Wendroff method is given by

$$\alpha_i = \frac{1}{2}w_i(1 + w_i\frac{\Delta t}{\Delta z}) + \frac{K_i}{\Delta z}, \quad \beta_i = \frac{1}{2}w_i(1 - w_i\frac{\Delta t}{\Delta z}) - \frac{K_i}{\Delta z}, \tag{2.22}$$

and non-dimensionalised by

$$a_i = \frac{1}{2}c_i(1+c_i) + s_i, \quad b_i = \frac{1}{2}c_i(1-c_i) - s_i. \tag{2.23}$$

From Appendix B.2.2 the conditions for positivity are

$$|c_i| \leq 2s_i + c_i^2, \quad i = 2, \ldots, N \tag{2.24}$$

$$s_i + s_{i+1} \leq 1 + \frac{1}{2}(c_i - c_{i+1}) - \frac{1}{2}(c_i^2 + c_{i+1}^2), \quad i = 2, \ldots, N-1. \tag{2.25}$$

The first condition can be restated as

$$P^{\text{cell}} \leq \frac{2}{1-|c_i|}. \tag{2.26}$$

With constant coefficients, the positivity conditions reduce to

$$|c| \leq c^2 + 2s \leq 1, \tag{2.27}$$

which is stronger than the stability condition

$$c^2 + 2s \leq 1. \tag{2.28}$$

There is no second order numerical diffusion in the transient solution. Wiggles will not occur if the positivity condition (2.26) is fulfilled. This method is implemented in the toolbox as the function `lwendrof`.

The Upstream Scheme

A non-centred alternative is the upstream method. This is the method used by Westgård (1989). The upstream scheme is studied in all books on the subject. The same diffusive flux (2.11) is used but the convective flux is estimated on the inflow side.

$$F_i^{\text{conv}} = w_i^+ \varphi_i + w_i^- \varphi_{i-1} \tag{2.29}$$

where

$$x^+ = \frac{1}{2}(x + |x|) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{2.30}$$

and $x^- = x - x^+$. In the notation of equation (2.8) the method is given by

$$\alpha_i = w_i^+ + \frac{K_i}{\Delta z}, \quad \beta_i = w_i^- - \frac{K_i}{\Delta z}, \tag{2.31}$$

and non-dimensionalised by

$$a_i = c_i^+ + s_i, \quad b_i = c_i^- - s_i. \tag{2.32}$$

From Appendix B.2.2 the positivity condition is simply

$$c_i^+ - c_{i+1}^- + s_i + s_{i+1} \leq 1. \tag{2.33}$$

With constant coefficients this reduces to

$$|c| + 2s \leq 1. \tag{2.34}$$

which is also the stability condition.

The numerical diffusion in the upstream scheme may be quite large

$$K_{num} = \frac{1}{2} w \Delta z (1 - |c|) \tag{2.35}$$

This is negligible if $K_{num} \ll K$, or equivalently

$$P^{\text{cell}} \ll \frac{2}{1 - |c|}. \tag{2.36}$$

The upstream scheme is implemented in the toolbox as the function `upstream`.

2.1.2 Nonlinear methods

The Lax-Wendroff scheme is second order in space but may develop wiggles and negative concentration values. The upstream method is positive but may be too diffusive. These are fundamental problems with linear schemes. Several nonlinear schemes has been developed to overcome these problems. A general idea is to combine the Lax-Wendroff and upstream methods to produce positive schemes with low numerical diffusion. Overviews of such methods are given by LeVeque (1992) and in the collection (Vreugdenhil and Koren, 1993).

Egg distribution problems are somewhat non-symmetrical. It is important to have high accuracy near maxima where most of the eggs are found. Local minima are less interesting and they occur more seldom in the interior of the water column. For instance, with the steady state solution (1.20) without source terms, a local minimum in the interior can occur only in static instable situations when the egg is lighter than the water above and heavier saline than the water below.

The methods below are not total variation diminishing (TVD) as some of the more advanced non-linear methods. Instead of a finely tuned linear combination, the schemes use simple on/off mechanisms to switch between the Lax-Wendroff and upstream fluxes. On the other hand, the asymmetry above is exploited. The schemes have high accuracy at maxima because the unmodified Lax-Wendroff flux is used in this situation.

The positive method

This is a simple scheme, mostly using the Lax-Wendroff fluxes but limiting to the upstream fluxes where Lax-Wendroff produces negative concentration values. The algorithm consists of three steps,

1. Compute the Lax-Wendroff fluxes including diffusivity by formulæ (2.21) and (2.11).
2. Apply the fluxes to compute a test distribution

$$\varphi_i^{test} = \varphi_i - \frac{\Delta t}{\Delta z}(F_i - F_{i+1}).$$

3. Where $\varphi_i^{test} < 0$, recompute F_i and F_{i-1} by the upstream formulation (2.29) and diffusion (2.11).

Positivity of the upstream scheme or Lax-Wendroff scheme implies positivity of the flux limited scheme. The positivity condition (2.33) is therefore a sufficient (but not necessary) condition for positivity of the combined scheme.

The scheme is implemented in the toolbox as the function `posmet`.

The minimum limiting method

Although the method above is positive, it may create wiggles around a positive value. To improve on this situation a slight variant is proposed. The new criteria for switching from Lax-Wendroff to upstream flux is the occurrence of a local minimum. As the first negative value must be a local minimum prevents the method from creating negative values (if the upstream scheme is stable).

Steps 1 and 2 in the algorithm are identical to the positive scheme. The third step is modified to

> 3'. Where $\varphi_i^{test} < \min(\varphi_{i-1}^{test}, \varphi_{i+1}^{test})$, recompute F_i and F_{i-1} by the upstream formulation (2.29) and diffusion (2.11).

This scheme is implemented in the toolbox as the function `minlim`.

2.1.3 The Source Term

Neglecting the fluxes for the moment, the transport equation (1.9) reduces to an ordinary differential equation

$$\frac{d\varphi}{dt} = P - \alpha\varphi. \tag{2.37}$$

The numerical scheme (2.7) reduces to

$$\varphi_i^{n+1} - \varphi_i^n = Q_i^n \Delta t. \tag{2.38}$$

The exact solution of (2.37) with constant coefficients is

$$\varphi^{n+1} = \mathrm{e}^{-\alpha\Delta t}\varphi^n + (1 - \mathrm{e}^{-\alpha\Delta t})\frac{P}{\alpha}. \tag{2.39}$$

This is in the form (2.38) with

$$Q_i = (P_i - \alpha_i \varphi_i) \frac{1 - \mathrm{e}^{-\alpha_i \Delta t}}{\alpha_i \Delta t}. \tag{2.40}$$

This scheme will never produce negative concentration values.

Combining this with the convection-diffusion solvers above is straightforward. The combined positivity condition becomes more restrictive as discussed in Appendix B.2.3. In general the number 1 appearing as upper limit must be replaced by $\exp(-\alpha\Delta t)$. For instance, with constant coefficients the positivity conditions for the familiar linear schemes become

FTCS	$\lvert c\rvert \le 2s \le \mathrm{e}^{-\alpha\Delta t}$,	(2.41)
Lax-Wendroff	$c^2 + 2s \le \mathrm{e}^{-\alpha\Delta t}$,	(2.42)
Upstream	$\lvert c\rvert + 2s \le \mathrm{e}^{-\alpha\Delta t}$.	(2.43)

2.2 The Stationary Problem

In many problems only the steady state solution is needed. Also for transient problems, a fast and accurate way of reaching the stationary solution is of interest.

If the coefficients w and K are constant, the exact solution (1.21) can be used directly. With negative velocity and low diffusivity, $m = w/K \ll 0$, the formula may overflow. This can be avoided by using the symmetry relation (1.22) for negative m. This solution is computed by the function `eggsact` in the toolbox. For the discretized solution to have the correct vertical integral and be comparable to the numerical solutions, cell averages

$$\varphi_i = \frac{\Phi}{\Delta z} \frac{\mathrm{e}^{mz_i} - \mathrm{e}^{mz_{i+1}}}{1 - \mathrm{e}^{-mH}} \tag{2.44}$$

are better. This is also computed by `eggsact`.

General w and K can be regarded as constant on the grid cells. Suppose w and K and their quotient m live on the flux points as in the transient problem. Form the cell average $\bar{m}_i = (m_i + m_{i+1})/2$. The cell averaged solution becomes

$$\bar{\varphi}_i = \frac{C_i}{\Delta z} \frac{\mathrm{e}^{\bar{m}_i z_i} - \mathrm{e}^{\bar{m}_i z_{i+1}}}{\bar{m}_i} \tag{2.45}$$

where the C_i-s are computed by continuity

$$\varphi_i(z_i) = \varphi_i(z_{i+1}) \tag{2.46}$$

and vertical integral

$$\Delta z \sum_{i=1}^{N} \bar{\varphi}_i = \Phi \tag{2.47}$$

With low mixing and high sinking velocity the exponential terms may overflow. The computation of the C_i-terms may overflow. The last problem is overcome by working with the logarithms. For the first problem the C_i-terms are renormalised by finding a suitable value for C_1. The method is implemented in the toolbox as the function `sstate`.

Chapter 3

Working in the MATLAB/Octave Environment

MATLAB and Octave offer command-line oriented interactive environments for numerical computations, data analysis and visualisation. They also provide high level programming languages. These programmes, called *M-files*, come in two flavours, *scripts* which are sequences of commands executed in batch, and *functions* which can take and return arguments. Functions can also have local variables. M-files may be viewed as extensions of the basic system. A collection of M-files for a specific area is called a *toolbox*. For marine research for example, there exists a free toolbox for analysis of physical oceanographic data, named SEAWATER (Morgan, 1994).

The basic environments and the programming languages are quite compatible. They provide the same basic data structure, arrays of dimension up to two (scalars, vectors, matrices). This makes it possible to write M-files that work equally well for both systems. The systems have simple but powerful systems for online help. The main difference between the systems is the much stronger graphic capabilities of MATLAB.

Octave is documented in (Eaton, 1995). MATLAB is documented in user and reference manuals, there is also a nice little primer (Sigmon, 1994).

3.1 Implementation of the VertEgg Toolbox

The geometric setup of the problem is basic for all other work. This setup is therefore given by a set of global variables which can be accessed in the work space and by all functions in the toolbox.

They are declared by the script `ve_init` which contain the following statements.

```
global Ncell   % Number of grid cells
global dz      % Vertical step size  [m]
global Hcol    % Depth of water column [m]
global ZE      % Ncell-vector of egg-point depths [m]
global ZF      % Ncell+1-vector of flux-point depths [m]
```

Given the depth `Hcol` $= H$ and the grid height `dz` $= \Delta z$, the rest of the values can be calculated. This is done by the function `ve_grid(Hcol,dz)`. The vector `ZE(i)` $= \bar{z}_i$ and `ZF(i)` $= z_i$ as defined in chapter ....

The commands `ve_init` and `ve_grid` can be reissued at any time. The only limitation is that `ve_init` must be called once before `ve_grid` and nearly all other VertEgg statements.

Many (but not all) of the commands have names starting with `ve_`, to not be mixed up with commands in the basic systems or other toolboxes. This may be done more consequently in future versions.

A discretized egg distribution is represented by a column vector of length `Ncell`. The elements are cell averages. A matrix of size `Ncell` $\times$ `n` can be viewed as a collection of `n` distributions. The commands in VertEgg and most general MATLAB/Octave commands work on distributions as a whole. For example, plotting the vertical profile is done by the command `plot(A,ZE)`.

The vector languages give a nice environment for working on 1D problems such as vertical distributions. For larger problems in 2D or 3D the limitation to 2D array[1] and the performance penalty of an interpreted language make MATLAB/Octave less interesting. Even for 1D problems there are certain tricks that must be used to gain acceptable performance. The most important is to use vector constructs instead of loops. For instance, to compute the central diffusive flux approximation (2.11) in Fortran 77 requires the loop.

```
      DO 11 I = 2, NCELL
        FDIFF(I) = - K(I) * (A(I-1) - A(I))
   11 CONTINUE
```

A similar loop is also possible in MATLAB/Octave, but much better performance is achieved by the vector subscript

```
Fdiff(2:Ncell) = - K(2:Ncell) * (A(1:Ncell-1) - A(2:Ncell)).
```

Under Octave, the variable `do_fortran_indexing` should be true. This gives better compatibility with MATLAB for vector subscripts.

[1]The restriction to 2D arrays is removed in MATLAB version 5

Chapter 4

Examples

This chapter contains several examples of different complexity on the use of the VertEgg toolbox. The examples are also available on line as demo scripts. Playing along and modifying these scripts is a nice way to learn to use the toolbox.

The online scripts come in two versions, one for MATLAB and one for Octave. This is partly due to differences in the languages, and partly to take advantage of the more advanced graphic possibilities in MATLAB. The examples are organized in separate directories. Issuing the command

```
help contents
```

in the right directory brings up an inventory of the scripts.

4.1 Example 1: The stationary solution with constant coefficients

This example demonstrates the use of `eggsact` and `srcsact` for computing the exact stationary solution in the case of constant eddy diffusivity and egg velocity with or without source terms. It also demonstrates how to make a simple vertical profile in both systems.

The script `eggsampl` demonstrates the use of `eggsact` First the vertical integrated concentration $\texttt{M} = 100\,\text{eggs/m}^2$, the eddy diffusivity $\texttt{K} = 0.01\,\text{m}^2\text{s}^{-1}$ and the egg velocity $\texttt{W} = 1\,\text{mm/s}$ must be defined.

```
M = 100;
K = 0.01;
W = 0.001;
```

The depth of the water column `Hcol` is set to 100 m, and a vertical grid size `dz` of 1 m is chosen.

```
ve_init
ve_grid(100,1)
```

With four arguments the function `eggsact(M,K,W,Z)` computes the solution (1.21) at a given depth level `Z`. Note that both positive and negative values can be used for the depth, `eggsact(M,K,W,Z)` = `eggsact(M,K,W,-Z)`.

With three arguments, `eggsact` computes the cell averages of the solution,

```
A = eggsact(M,K,W);
```

The toolbox contains functions that compute the integral, mean depth and standard deviation of `A`.

```
ve_int(A)
ans = 100.0
ve_mean(A)
ans = -9.9975
ve_std(A)
ans = 9.9773
```

By construction the integral is correct. Using formulas (1.27) and (1.28), the correct values for the mean and standard deviation are computed by

```
m = W/K;
mean = - (1 / m) + H / (exp(m*Hcol) - 1)
mean = -9.9955
var  = (2-exp(-m*Hcol) * (m^2*Hcol^2 + 2*m*Hcol + 2)) / ...
              (m^2*(1-exp(-m*Hcol))) - mean^2;
std = sqrt(var)
std = 9.9773
```

The profile of the egg distribution `A` can be plotted by the command,

```
plot(A,ZE)
```

a nicer plot as shown in figure 4.1 is made under MATLAB by the sequence,

```
plot(A,ZE)
title('Stationary solution')
xlabel('Concentration [eggs/m^3]')
ylabel('Depth [m]')
```

under Octave the `plot` command terminates the sequence.

```
title('Stationary solution')
xlabel('Concentration [eggs/m^3]')
ylabel('Depth [m]')
plot(A,ZE)
```

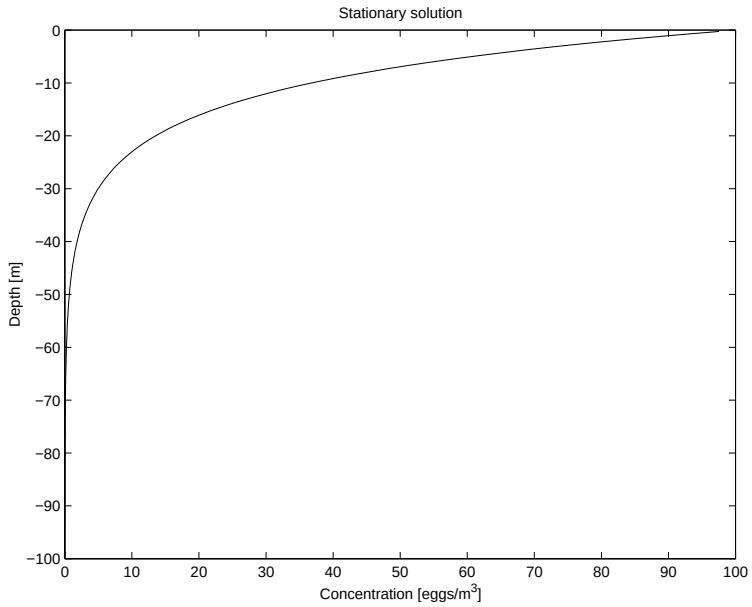

Figure 4.1: Steady state solution with constant coefficients and no source terms

.

The script `srcsampl` demonstrates the exact solution `srcsact` with source terms. Using the same values of `K` and `W` and spawning rate $1\,\mathrm{egg/m^3/day}$. and mortality of 5% per hour. The mortality rate `alpha` is computed by $exp(-\alpha\Delta t) = 1 - 0.05$. The integrated steady state concentration in this case is $\alpha/(P * H) = 81.2\,\mathrm{eggs/m^2}$.

The solution is computed by the following code segment

```
K = 0.01;
W = 0.001;
P = 1/(24*3600);
alpha = -log(0.95) / 3600;
A = srcsact(K,W,P,alpha);
```

The steady state solution `A` is depicted in figure 4.2. The main difference from the nosource situation is that the concentration stays well above zero except very close to the bottom, where no eggs come up from below.

4.2 Example 2: Transient solution with constant coefficients

This example demonstrates the use of `ftcs`, `lwendrof` or `upstream` and `fluxlim` for solving the stationary problem with constant values of eddy diffusivity and egg velocity. In addition more visualisation techniques including animation is demonstrated. The online scripts for

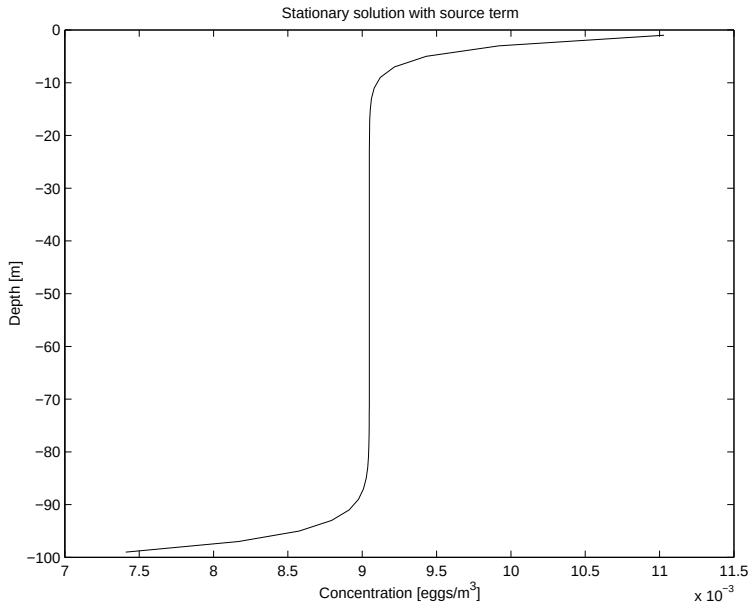

Figure 4.2: Steady state solution with constant coeffients including spawning and mortality

this example is `runftcs`, `runus`, `runlw`, `runfl`, `plotex2` and `animex2`. Of these one of the `run`-scripts must be run first, to produce the data for the visualization scripts.

The model is set up by giving values to the defining parameters.

```
H = 100;        % Depth [m]
dz0 = 2;        % Grid size [m]

dt = 120;       % Time step [s]
outstep = 1;    % Time between saving of model results [hours]
simtime = 96;   % Total simulation time [hours]

K0 = 0.01;      % Eddy diffusivity [m^2/s]
W0 = 0.001;     % Egg velocity     [m/s]

M0 = 1000;      % Vertical integral of concentration [eggs/m^2]
```

The space variables is used to initialize the grid for VertEgg.

```
ve_init
ve_grid(H,dz0)
```

Further time variables are needed

```
nstep  = outstep*3600/dt; % Number of steps between outputs
nout   = simtime/outstep; % Number of output steps
```

To make a start distribution concentrated at 50 m use the `spawn` function

```
A0 = spawn(M0, 50);
```

alternatively a random initial condition can be given by `A0 = ve_rand(M0)`.

As the solvers need vectors as input, `K0` and `W0` must be converted into constant column vectors.

```
K = K0*ones(Ncell+1,1);
W = W0*ones(Ncell+1,1);
```

We will use `A` for the calculations and `X` for saving the results.

```
A = A0;
X = [];   % Remove any old stuff in X
```

The time integration loop can be written as

```
for t = 1:nout
  A = lwendrof(A,K,W,nstep,dt);
  X(:,t) = A;
end
```

After computing the solution, the results must be analysed and visualised. The final result from `runlw` is stored in `A`. The following code-fragment plots the profile of `A` and the exact stationary solution `B`.

```
plot(A,ZE)
hold on   % Don't wipe out graphics before next plot
B = eggsact(M0,K0,W0);
plot(B,ZE,'r')
hold off
legend('Numerical solution', 'Exact solution')
```

For Octave remove the `legend` statement.

The convergence of the solution to a steady state can be studied by looking at the time development of the mean.

```
T  = [1:nout];
TL = T*outstep;          % Time labels
mu = ve_mean(X);         % The means
plot(TL,mu)
```

An alternative is to look at the root mean square deviation from the exact solution `B`. (!!! Sjekk om her er noe semilog-greier)

```
R = ve_rmsd(X,B);
plot(TL,R)
```

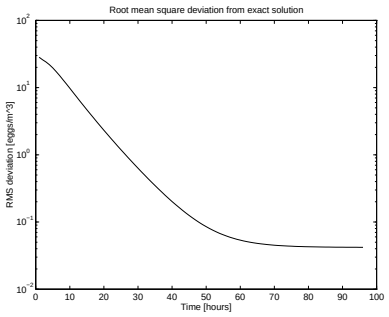

Figure 4.3: Convergence to steady state with Lax-Wendroff method

By removing the first 24 hours, the convergence is easier to see. This result is depicted in figure 4.3

```
T2  = [24:nout];
TL2 = T2 * outstep;
plot(TL2, R(T2));
```

The two next figures are MATLAB only. Surface graphics in Octave/gnuplot is to primitive yet. The first is a isoplet diagram, a contour plot of the solution in the time-depth plane.

```
cl = [10:10:110];  % Contour levels
c = contour(TL,ZE,X,cl);
clabel(c)
```

A more spectacular view of the same surface in 3D.

```
surf(TL,ZE,X);
```

The last script `animex2` animates the time evolution of the numerical solution. Under MATLAB be careful to make the graphic window visible and disjoint from the command window, otherwise the command window will be raised and hide the animation. The MATLAB animation code is simply.

```
T = [1:nout];
```

```
for t = T
  title_string = sprintf('Time = %d', t*outstep)
  plot(X(:,t),ZE)
  title(title_string)
  axis([0 120 -Hcol 0])
  drawnow
end
```

The Octave code is nearly the same.

```
T = [1:nout];
axis([0 120 -Hcol 0])
for t = T
  title_string = sprintf('Time = %d', t*outstep);
  title(title_string);
  plot(X(:,t), ZE)
end
```

If the animation is too fast, put the statement `pause(1)` somewhere in the loop.

4.3 Example 3, Sensitivity studies

In this example, the performance of the different numerical schemes from section 2.1 is tested. The testing is done by the script `runsens`. The testing is done by running the schemes with constant coefficients until steady state and thereafter comparing with the exact solution computed by `eggsact`. The parameters that are varied are the eddy diffusion coefficient K, the vertical velocity w, the space step `dz` and the time step `dt`. These are defined near the start of the file `runsens.m`.

```
% Diffusion coefficient [m^2/s]
K0 = 0.01;
% Vertical velocity     [m/s]
W0 = 0.001;
% Space step            [m]
dz0 = 2;
% Time step             [s]
dt = 120;
```

These values are the same as in example 2. The other defining variables are

```
H = 100;       % Depth [m]
simtime = 72;  % Simulation time [hours]
M0 = 1000;     % Vertical integral of concentration [eggs/m^2]
```

From example 2 it is evident that the result after 72 hours is close to the steady state limit. All eggs are released at depth 50 m.

After initialising VertEgg, the characteristic numbers are computed

```
C     = W0 * dt/dz;
S     = K0 * dt/(dz*dz);
Pcell = abs(C)/S;
```

they are printed nicely in the command window by the C-style output function `fprintf`.

The number of timesteps is

```
nstep  = simtime*3600/dt;  % Number of steps
```

Testing the first scheme is done by

```
tic           % Reset clock
A = ftcs(A0,K,W,nstep,dt);
tid(1) = toc; % Save elapsed time
X(:,1) = A;   % Put solution as first column in X
```

and similar for the other three methods. The timing was done on a PC with an INTEL 486 processor running at 66 MHz.

The exact solution and some of it properties is computed by

```
B = eggsact(M0,K0,W0);
Bmean = ve_mean(B);
Bstd  = ve_std(B);
```

Thereafter the root mean square error, and the errors in mean depth and standard deviation is computed by

```
R = ve_rmsd(X,B);
Emean = ve_mean(X) - Bmean;
Estd  = ve_std(X) - Bstd;
```

For plotting purposes, the errors in the methods are given as the columns in `Y`.

```
Y = X - B *ones(1,4);
```

The results are summarized in a table, made by the `fprintf` command. A better formatted version of this table for a standard run is given in table 4.1.

While performing such sensitivity tests under MATLAB the `diary` function can be useful. It saves the commands and responces in the text window to a file for later use.

The tests presented here belong to two groups. The first has $K = 0.01\,\mathrm{m^2 s^{-1}}$ and $W = 0.001\,\mathrm{ms^{-1}}$, the same values as used in previous examples. The exact stationary solution with `dz` $= 1$ is given as fig 4.1.

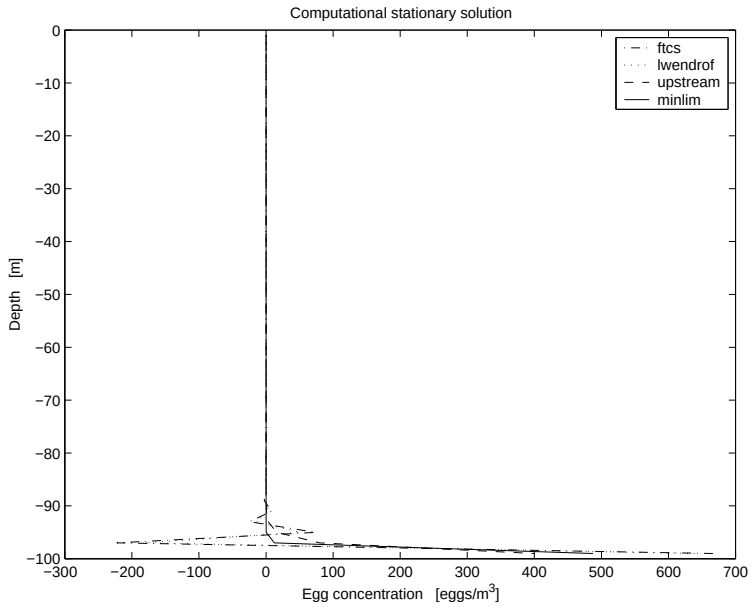

Figure 4.4: The numerical stationary solutions with parameters, $K = 0.01\,\mathrm{m^2 s^{-1}}$, $w = 0.001\,\mathrm{ms^{-1}}$, `dz` $= 2\,\mathrm{m}$, `dt` $= 120\,\mathrm{s}$

The reference run is done by with a space step of 2 mand a time step of two minutes. The resulting solutions are given in fig. 4.4. All methods, except the upstream scheme, produced solutions that can not be distinguished from each other (and the exact solution) in the figure. The upstream scheme gives a lower peak at the surface and too high concentrations below 10 m. Figure 4.5 plots the error, that is the diffence between the numerical solution and the exact solution. As Lax-Wendroff is positive, the flux limited method produce the same solution. The FTCS scheme slightly overshoots the surface value, while Lax-Wendroff undershoots even more slightly. The table for this run is table 4.1 below.

	RMSE	E-mean	E-std	CPU-time
ftcs	0.050	0.029	-0.026	8.7
lwendrof	0.044	-0.030	0.034	8.5
upstream	1.421	-0.965	0.951	8.6
fluxlim	0.044	-0.030	0.034	11.8

Table 4.1: Results from reference run

The other class of tests is meant to be typical of sinking eggs. Here the eddy diffusivity is 5 per cent of the above, $K = 0.0005\,\mathrm{m^2 s^{-1}}$ and $w = -0.001\,\mathrm{ms^{-1}}$. To plot the exact solution for a subrange of the water column, an index array is used.

```
B = eggsact(M0,K0,W0);
```

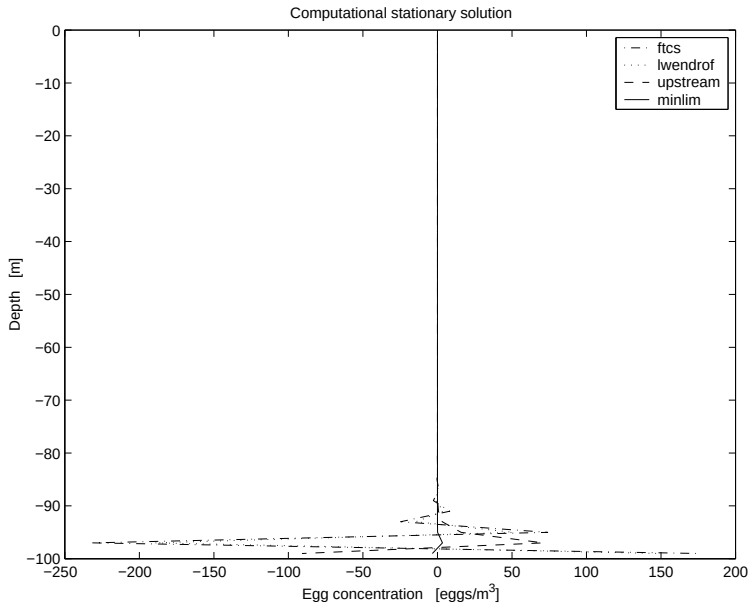

Figure 4.5: The errors in the numerical stationary solutions above

```
II = 41:50;
plot(B(II),SE(II))
```

This figure is given as figure 4.6.

The corresponding numerical solutions (`dz` = 2 m, `dt` = 120 s) are shown in figure 4.7. Both the FTCS and the Lax-Wendroff scheme have produced oscillations and negative values. The upstream solution is quite smeared out, while the flux limited method seem to perform quite well. The table for this run is tab 4.2.

	RMSE	E-mean	E-std	CPU-time
ftcs	42.549	-0.537	-0.640	8.6
lwendrof	35.038	-0.477	-0.640	8.5
upstream	16.466	-0.463	0.618	8.6
fluxlim	0.718	-0.014	-0.022	32.8

Table 4.2: Results from standard sinking run

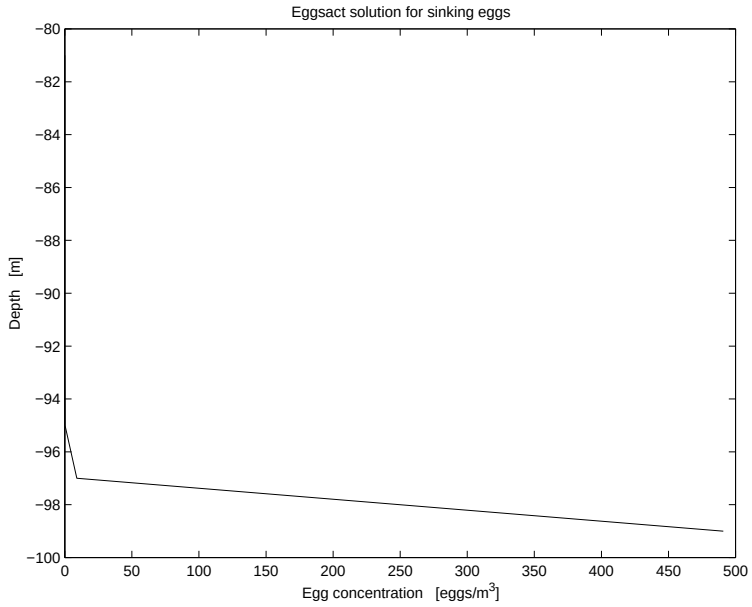

Figure 4.6: The exact stationary solution with$K = 0.0005\,\mathrm{m^2 s^{-1}}$, $w = -0.001\,\mathrm{ms^{-1}}$,`dz` = $2\,\mathrm{m}$

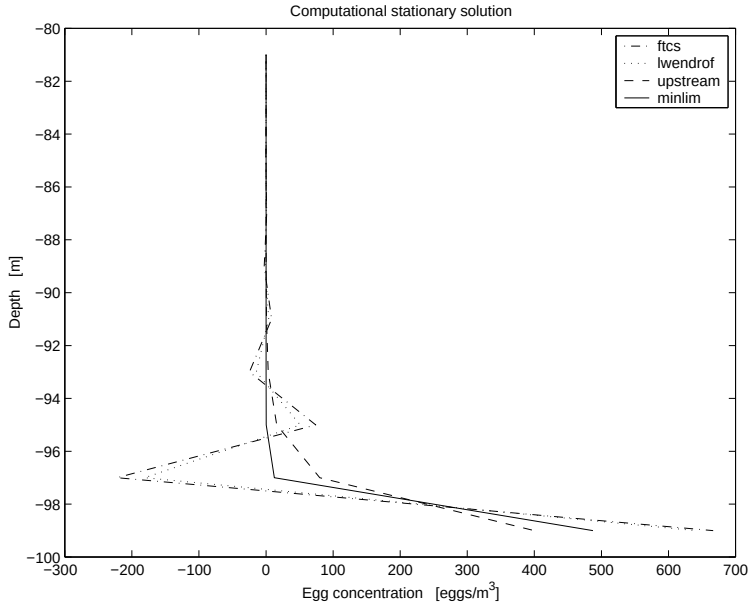

Figure 4.7: The numerical stationary solutions with parameters, $K = 0.0005\,\mathrm{m^2 s^{-1}}$, $w = -0.001\,\mathrm{ms^{-1}}$, `dz` = $2\,\mathrm{m}$, `dt` = $120\,\mathrm{s}$

4.4 Example 4, Terminal egg velocity

This example verifies and demonstrates the function `eggvel` for computing the terminal egg velocities. The script is `velex` in the example directory. Incluided in this example is also the script `visklsq` which performs the least squares regression used to derive formula (1.47) for the dynamic molecular viscosity.

First some necessary constant are defined,

```
mu = 1.6e-3;          % Dynamic molecular viscosity
g = 9.81;             % Acceleration due to gravity
rho = 1027;           % Density of sea water
```

then the maximum egg diameter `Dmax` for application of Stokes' formula can be plotted

```
drho = [0.1:0.1:5];      % Range of density differences
Dmax = ( 9 * mu^2 ./ (rho * g * drho) ).^(1/3);
plot(drho, Dmax * 1000)  % Use mm as unit
```

As a verification of the implementation of the formulas in `eggvel`, figure 1 from Sundby (1983) will be reproduced.

```
d = [0:0.1:5];   % Range of diameters (in mm)
hold on
axis([0 5 0 5]);
for drho = [0.25 0.5 1:6]  % The values of drho used by Sundby
  W = eggvel(drho * ones(size(d)), d/1000, mu);
  plot(d, W*1000, 'g')
end
```

The expression `drho*ones(size(d))` is required to make `drho` into an array of the same shape as `d`, as required by `eggvel`. The factors 1000 converts between mm and m. The lines $Re = 0.5$ and $Re = 5.0$ are added in red colour. This figure is shown as fig. 4.8.

```
d(1) = [];          % Remove d(1) to prevent division by zero
W = 0.5 * mu ./ (rho * d/1000);
W = 1000 * W;       % Convert to mm/s
hold on
plot(d, W, 'r');    % Re = 0.5
plot(d, 10*W, 'r'); % Re = 5.0
hold off
```

Another way of visualizing the `eggvel` function is to make a table and make a contour plot of it. The rest of this example is for MATLAB only. Tables for `W` and `Re` are made by the following commands,

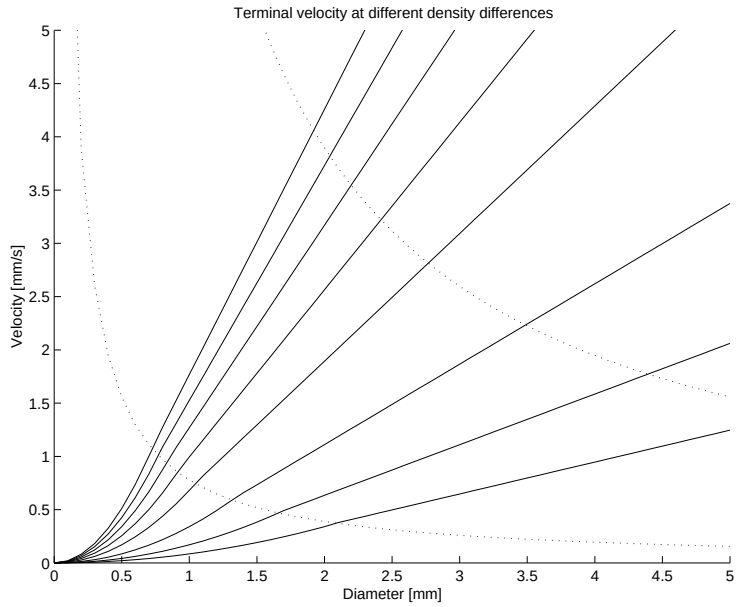

Figure 4.8: Terminal velocity for different density differences

```
d = [0:0.1:4];      % mm
drho = [0:0.2:6];   % kg/m^3
dtab = ones(size(drho))' * d / 1000;  % d constant in colums
drhotab = drho' * ones(size(d));  % drho constant in rows
[W Re] = eggvel(drhotab, dtab);
```

The contour plot is made by.

```
cl = [0.1:0.1:0.5 1.0:0.5:3.0 4.0:9.0]; % Contour levels
c = contour(d, drho, 1000*W, cl, 'g');
clabel(c)
```

Add the same lines $Re = 0.5$ and $Re = 5.0$ in red as above.

```
hold on
cl = [0.5 5.0];
contour(d, drho, Re, cl, 'r');
hold off
```

A 3D view of the velocity surface can also be plotted.

```
surf(d, drho, 1000*W)
```

Systems like MATLAB and Octave are well suited for fitting functions to data. The script `visklsq` demonstrates how to fit a polynomial, quadratic in T and linear in S to

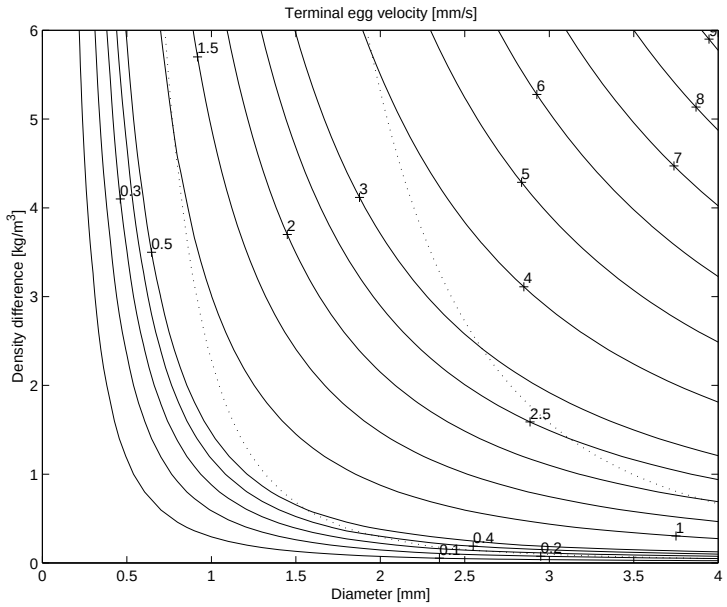

Figure 4.9: Terminal velocity contours

table 1.1 by the method of least squares. Some array manipulation must first be done to obtain three column vectors `SI`, `TI` and `B` with corresponding values of salinity, temperature and viscosity. With `MN` $= 35 = 5 \times 7$, the normal matrix is simply

```
A = [ones(MN) TI  TI.^2 SI];
```

In MATLAB the vector `X` of coefficients is found by

```
V = eye(MN);        % Covariance matrix = identity
X = lscov(A,B,V);
```

in Octave the same task is done by

```
X = ols(B,A);
```

4.5 Example 5, Non-constant coefficients

This example illustrates how to use several egg groups on a transient problem with non-constant coefficients. It also provides a verification of parts of the toolbox by reproducing results by Westgård (1989). The model is contained in the script `runex5`. The results are visualized by the scripts `plotex5` and by animation in `animex5`.

The problem is the first example from Westgård (1989). The bottom depth is 100 m and a grid size of 2 m is used.

```
ve_init
ve_grid(100,2)
```

There is a pycnocline at 50 m, with temperature 7°C and salinity 34 above and temperature 5°C and salinity 35 below.

```
I25 = ones(25,1);
S(1:25) = 34*I25; S(26) = 34.5; S(27:51) = 35*I25;
T(1:25) =  7*I25; T(26) =  6  ; T(27:51) =  5*I25;
```

The wind speed is $5\,\mathrm{ms}^{-1}$. The turbulent eddy viscosity in the upper layer is computed by a formula form Sundby (1983). The turbulence in the lower layer is 1/10 of the value in the upper layer.

```
Wind = 5;
K0 = (76.1 + 2.26*Wind^2) * 1e-4;
K(1:25) = K0*I25; K(26) = 0.55*K0; K(27:51) = 0.1*K0*I25;
```

There are two groups of eggs, one with a diameter of 1.5 mm and neutral buoyancy corresponding to 33 psu, and the other with a diameter of 1.3 mm and neutral salinity 36 psu. The tool `eggvelst` is used to compute the velocity profiles.

```
d1  = 0.0015;
Se1 = 33;
W1  = eggvelst(S, T, d1, Se1);
d2  = 0.0013;
Se2 = 36;
W2  = eggvelst(S, T, d2, Se2);
```

The number of eggs are 55 in each group and the initial distribution is the same step function in each group. The arrays `A1` and `A2` are used for the concentrations within the groups.

```
I10 = ones(10,1);
A1( 1:10) = 0*I10;
A1(11:20) = 0.75*I10;
A1(21:30) = 1.25*I10;
A1(31:40) = 0.75*I10;
A1(41:50) = 0*I10;
% Use the same initital distribution for group 2
A2 = A1;
% Save the intial distributions for later use.
A10 = A1; A20 = A2;
A0 = A10 + A20;
```

The model is here run by the Lax-Wendroff scheme with a time step of 120 s and saving the result every 2nd hour. The total simulation time is 96 hours.

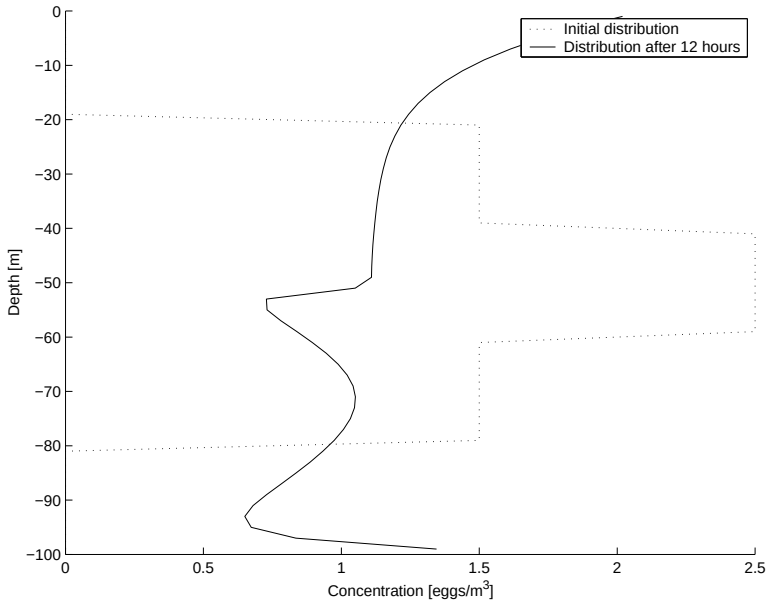

Figure 4.10: Figure 1 from Westgård (1989)

```
for t = 1:nout
  A1 = lwendrof(A1,K,W1,nstep,dt);
  X1(:,t) = A1;
  A2 = lwendrof(A2,K,W2,nstep,dt);
  X2(:,t) = A2;
end
```

Finally the results from the two groups are added,

```
X = X1 + X2;
```

Visualising the results with `plotex5` the first task is to reproduce figure 1 from Westgård (1989).

```
t12 = 12 / outstep;
hold on;
plot(A0, ZE, 'y');        % The start distribution
plot(X(:,t12), ZE, 'g');  % After 12 hours
```

This result is depicted in Fig. 4.10.

With non-constant coefficients the function `sstate` must be used to calculate the stationary solution.

```
Y1 = sstate(M1,K,W1);
Y2 = sstate(M2,K,W2);
Y = Y1 + Y2;
```

Adding this to the previous figure, gives Fig. 4.11.

```
plot(Y, ZE, 'r');
```

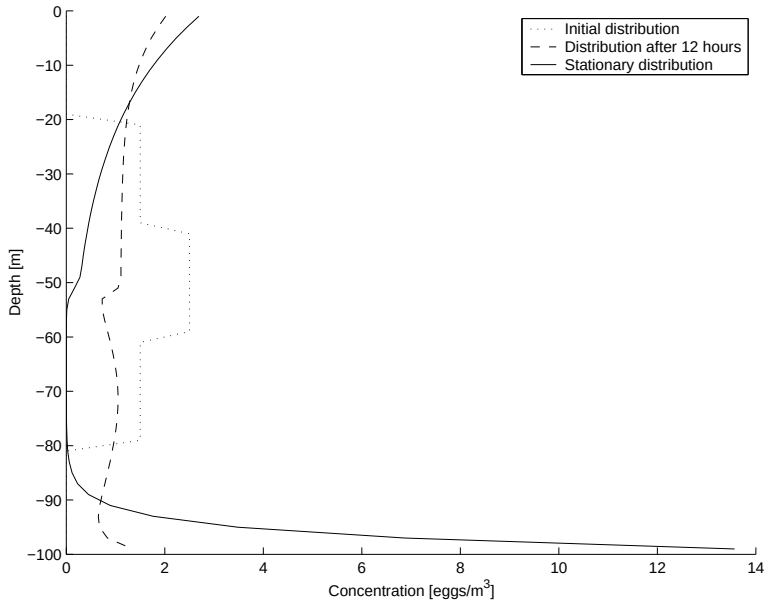

Figure 4.11: Initial, 12 hour and stationary limit solution

In addition `plotex5` and `animex5` show more details on the time evolution. For this example the steady state solution is reached after approximately 80 hours.

4.6 Example 6, Halibut eggs, the transient problem

In this and the next two examples, eggs of Atlantic halibut *(Hippoglossus hippoglossus)* will be considered in deeps fjords in northern Norway. Data on egg diameter and egg salinity are taken from (Haug et al., 1984). Synthetic vertical profiles, representative for the area, of temperature, salinity and vertical mixing have been constructed by S. Sundby (pers. comm.).

This example goes further towards developing a model application from VertEgg. The main script `runex6` is initiated from the set-up file `runex6.sup`. The physical setting and initial egg distribution are also read from files. The model script is meant to be a prototype simulation model. It should be easy to modify for similar simulations. The script `plotini` is used to plot the physical and numerical situation before the simulation, while `plotgrp3` and `plotres` are used for postprocessing of the model results.

First the physical profiles will be plotted. The profiles are read from file and thereafter interpolated to the flux points. The variable names are `S` for salinty, `T` for temperature

and K for vertical mixing coefficient. Several profiles are combined into one figure by the Matlab command subplot.

```
subplot(1,3,1)
plot(S,ZF)
title('Salinity')
subplot(1,3,2)
plot(T,ZF)
title('Temperature')
subplot(1,3,3)
semilogx(K,ZF)   % Logarithmic X-axis
title('Vertical mixing')
```

The result is shown in fig. 4.12

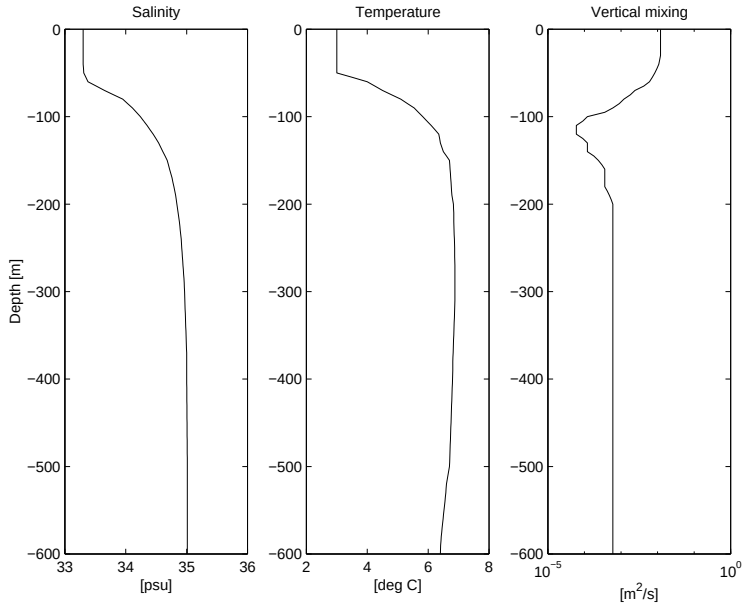

Figure 4.12: Vertical profiles of salinty, temperature and vertical eddy viscosity

With the values 34.5 psu for egg salinity and egg diameter 3.2 mm, eggvelst can be used to compute the egg velocity. Thereafter the stationary solution with integrated density 100 eggs/m^3 can be computed by sstate.

```
eggsal = 34.5;
eggdiam = 3.2;
W = eggvelst(S,T,eggdiam/1000.0,eggsal);
M = 100;
ASS = sstate(M,K,W);
```

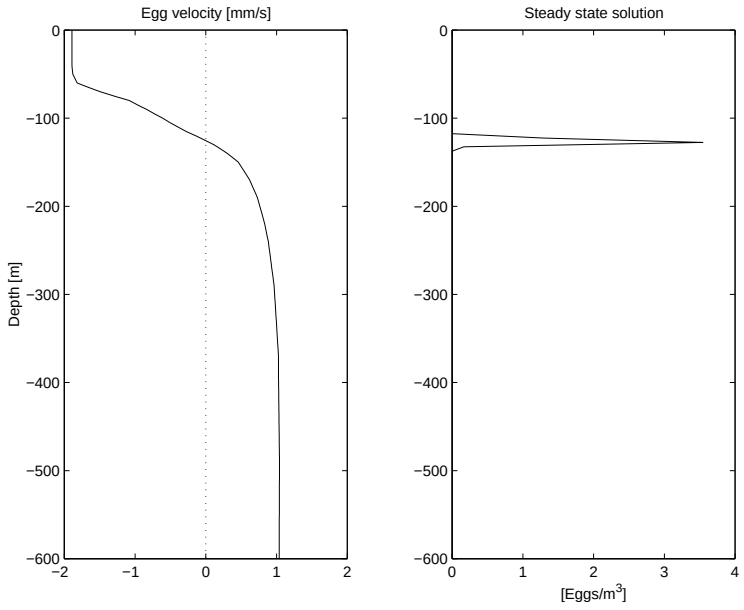

Figure 4.13: Egg velocity and stationary ditribution

The profiles are plotted in figure 4.13. The vertical dotted line in the left indicate zero velocity.

The neutral salinity 34.5 occur at approximately 125 m depth, in the pycnocline layer where the vertikal mixing coefficient is very low. The stationary solution is therefore a very narrow peak at this depth level.

Before running the transient simulation it is useful to look at the numerical characteristics of the problem. With $\Delta z = 5\,\mathrm{m}$ and $\Delta t = 600\,\mathrm{s}$ the profiles are plotted in figure 4.14. All the three linear methods are stable. However, the methods are not suitable because the high Peclet numbers give high numerical diffusion in the upstream method and produce negative values with FTCS and Lax-Wendroff. Therefore the more time-consuming method `posmet` is used for the transient simulations.

The model script start by reading the setup-file. Thereafter the vertical physical profiles are read in and interpolated as above. Similarly the inital egg distribution is read from file `halibut.dat` and interpolated to the egg points.

Initially the eggs are distributed between 500 and 600 m depth. The model is run with space step $\Delta z = 5\,\mathrm{m}$ and time step $\Delta t = 10\,\mathrm{minutes}$ as above. 5 egg groups are used, all with diameter 3.2 mm but the neutral salinity vary from 34.3 to 34.7 psu in steps of 0.1. The simulation time is 10 days, and the results are written to file every 12 hours.

The first post-processing script `plotgrp3` concentrates on egg group 3, which is neutral at salinity 34.5 psu. The model output file `result.dat` is read and the egg profile data every 24 hour starting with the initial distribution is stored in the array `X3`. The command

```
plot(X3,ZE)
```

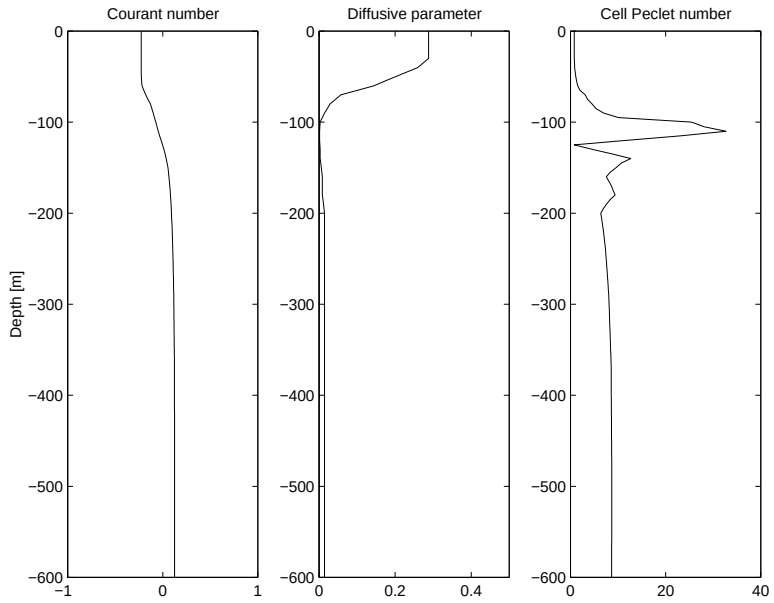

Figure 4.14: Vertical profiles of Courant number, diffusive parameter and cell Peclet number

plots these daily distributions in fig. 4.15. The numerics seems to be working, the distributions do not contain negative values and the eggs are moving upwards a little less then 100 m per day without too much smoothing of the distribution. After 5–6 days the distribution has reached it equilibrium level and start to concentrate at this level. After the 10 days, the transient solution is very similar to the stationary solutions. These solutions are compared in figure 4.16. The transient solution have a slightly higher peak consentration.

The script `plotres` is used for postprocessing the results for all 5 egg groups. In this case the columns in `X` hold the sum of the 5 concentrations every 12 hour. Figure 4.17 show the distributions for the 5 egg groups after the 10 days of simulation. The solution is a narrow peak for each egg group at their equilibrium salinity.

The next figure 4.18 show the time evolution of the mean depth of the distributions. Note that although the distance is shorter the heaviest eggs use longer time to raise to their equilibrium depth. This confirms unpublished calculations by S. Sundby.

4.7 Example 7, Halibut eggs, Monte Carlo simulation

Under construction

In nature, egg diameters and densities are not constant or confined to a handful of discrete groups. To explore this further it is assumed that these properties have known statistical distributions. The paper (Haug et al., 1984) does not identify the distributions but give ranges for the values. For the egg diameter a normal distribution with mean

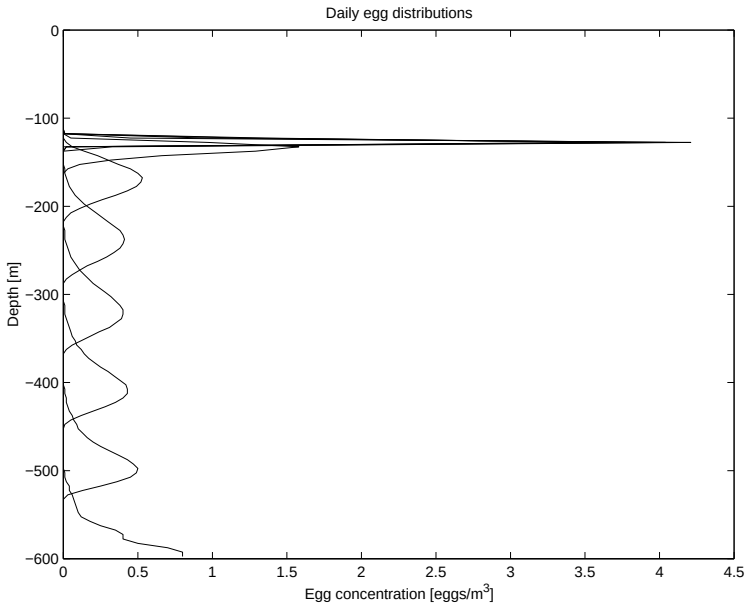

Figure 4.15: Daily egg distributions from egg group 3

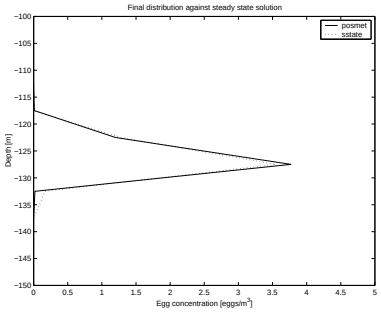

Figure 4.16: The distribution after 10 days compared to the steady state distribution from `sstate`

3.2 mm and standard deviation 0.2 mmis used below. For the neutral salinity of the egg a normal distribution with mean 34.5 psu and standard deviation 0.2 psu is used. For simplicity it is assumed here that egg diameter and egg salinity are independent.

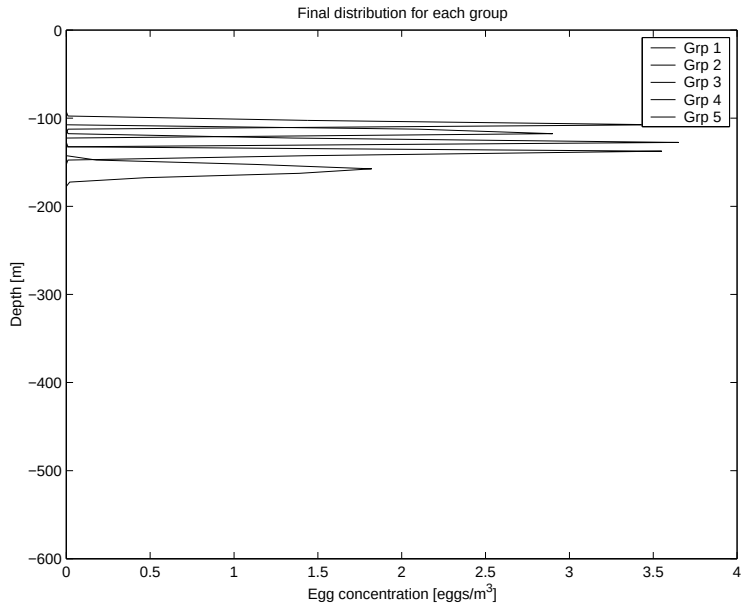

Figure 4.17: Final distribution of the 5 egg groups

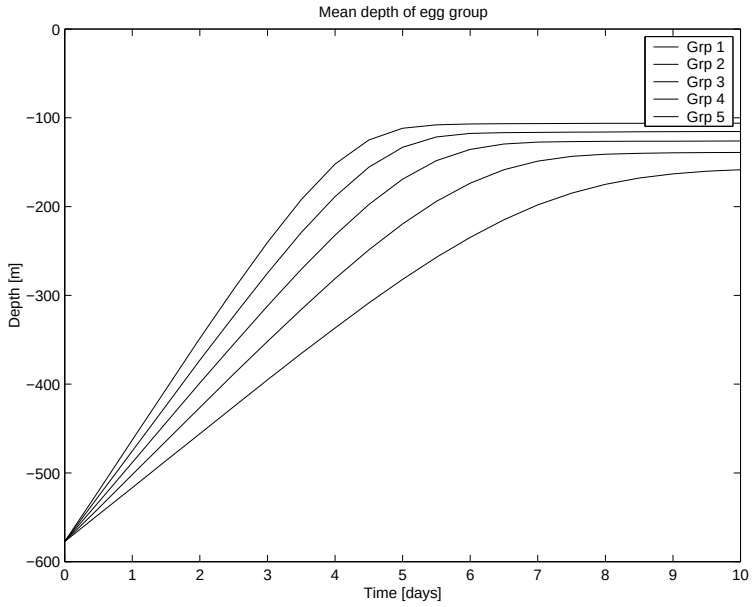

Figure 4.18: Time evolution of mean depth

In the script `runex7` a total of `Nprof` egg diameters and corresponding egg salinities are drawn randomly from the distributions above. Thereafter `eggvelst` is used to compute the velocity profiles and `sstate` computes the vertical distribution. The variable `Anorm`

contains the average of the distributions so far, normalised so that the vertical integral equal `M` eggs/m^2.

The script `plotex7` is used to plot the results. With `Nprof` = 1000 the averaged vertical profile is given in figure 4.19. Nearly all the eggs are found between 50 and 300 m depth with high concentrations between 100 and 200 m. Also note the few heavy eggs are near bottom.

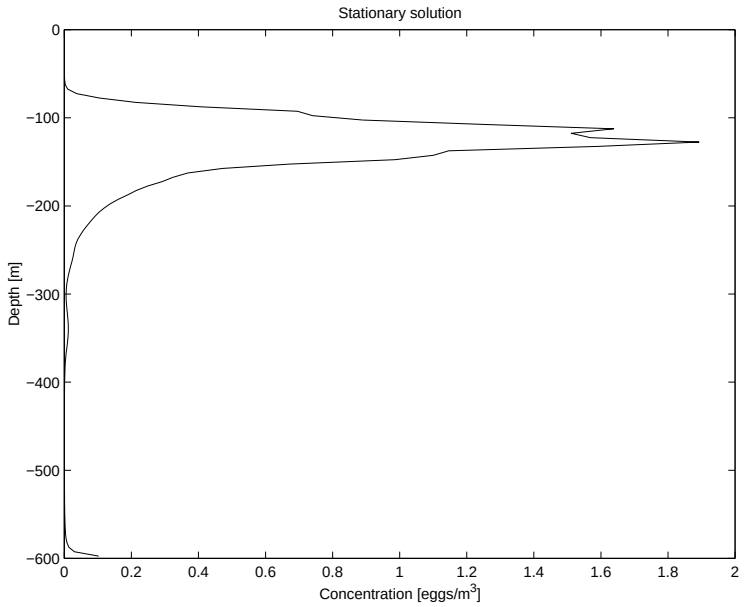

Figure 4.19: Vertical distribution of eggs from Monte Carlo run

Thereafter some statistics of this solution is computed. Because of the randomness in the Monte Carlo process these numbers (and the shape of fig. 4.19) may vary from run to run. In our case we obtain the following numbers. The egg diameters vary from 2.57 to 3.85 mm with mean 3.19 mm and standard deviation 0.21 mm. The egg salinities vary from 33.87 to 35.22 with mean 34.50 and standard deviation 0.20 psu. The mean depth of the distribution is 135.1 mand the standard deviation is 50.7 m.

To visualize the convergence of the Monte Carlo run, the root mean square deviation of the normalised distribution after $i-1$ and i iterations are saved in the array $R(i)$. Figure 4.20 show the evolution of R on a logarithmic scale. The convergence is quite slow, and 1000 samples may be too few. This explains why the shape of the distribution may vary from run to run.

4.8 Example 8, Halibut eggs with spawning and hatching terms

Under construction

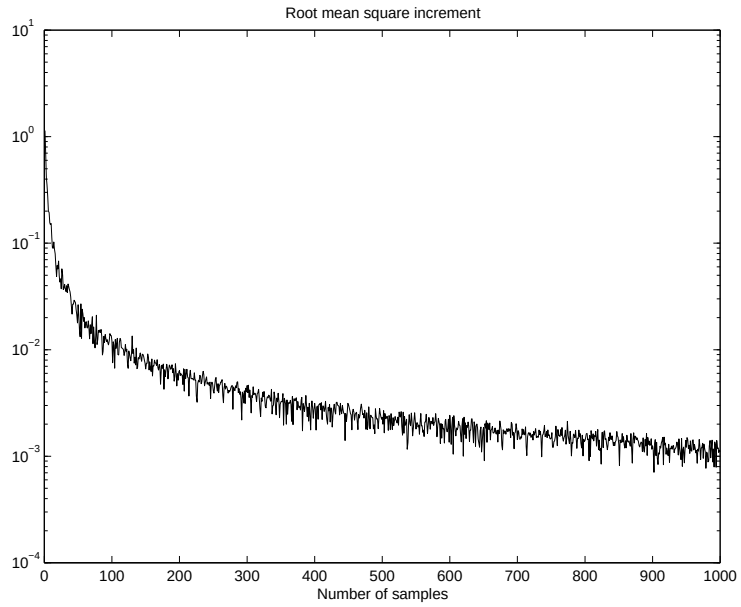

Figure 4.20: Development of the root mean square increment

Chapter 5

Reference Manual — VertEgg Version 0.9

5.1 Overview of the Toolbox

The VertEgg toolbox presently contain the following tools, grouped here according to functionality.

- Initialise VertEgg
  - `ve_init`
  - `ve_grid(Hcol, dz)`
- Make initial egg distribution
  - `spawn(M, Z)`
  - `ve_rand(M)`
- Compute the stationary solution
  - `eggsact(M, K, W, Z)`
  - `srcsact(K, W, P, alpha)`
  - `sstate(M, K, W)`
- Solve the transient problem numerically
  - `fluxlim(A0, K, W, nstep, dt, P, alpha)`
  - `ftcs(A0, K, W, nstep, dt, P, alpha)`
  - `lwendrof(A0, K, W, nstep, dt, P, alpha)`
  - `upstream(A0, K, W, nstep, dt, P, alpha)`

- Compute terminal egg velocity
  - `dens0(S, T)`
  - `eggvel(drho, d, mu)`
  - `eggvelst(S, T, d, Se)`
  - `molvisc(S, T)`
- Analyse distributions
  - `eggmom(A, p)`
  - `ve_drint(A, z1, z2)`
  - `ve_int(A)`
  - `ve_mean(A)`
  - `ve_std(A)`
  - `ve_rmsd(A, B)`

5.2 Description of the Tools

dens0 – Sigma-T of sea water at zero pressure

Usage: `sigma = dens0(S, T)`

Input:

`S`	: Salinity	[psu]
`T`	: Temperature	[°C]

`S` and `T` may be arrays of the same shape.

Output:

`sigma`	: Sigma-T value	$[\,\mathrm{kgm}^{-3}]$

`sigma` is an array of the same shape as `S` and `T`.

Description:

`dens0` computes the σ_T value (density - 1000) of sea water at zero pressure. The density is computed by the international equation of state for sea water, UNESCO, 1980.

eggmom – Moment of egg distribution

Usage: `M = eggmom(A, p)`

Input: `A` : Egg distribution [eggs/m^3]
`p` : Order of moment

`A` must be a matrix where the columns live on egg-points, but a row-vector at egg-points is also accepted, `size(A) = (Ncell x n)` or `(1 x Ncell)`

Output: `M` : The `p`-th moment of `A` [eggs m^{p-2}]

If `A` is a matrix of size `(Ncell x n)`, `M` becomes a row-vector of length n containing the moments of the columns of `A`. `p` must not be negative.

Description:

Computes the `p`-th moment of the egg-distribution `A`,

$$M = \int_{-H}^{0} z^p a(z) dz$$

where $a(z)$ is the piecewise constant function $a(z) =$ `A(i)` for `ZF(i+1)` $< z <$ `ZF(i)`. If `A` is a matrix, the moments of the columns are calculated.

eggsact – Exact stationary solution, const. coeff.

Usage: `A = eggsact(M, K, W, Z)`

Input: `M` : Vertical integrated concentration [eggs/m^2]
`K` : Eddy diffusivity [m^{2}s^{-1}]
`W` : Terminal velocity [ms^{-1}]
`Z` (opt) : Vertical coordinate [m]

`M`, `K`, `W` are scalars. `Z` can be arbitrary array. IF `Z` is omitted, `ZE` is used as vertical coordinates.

Output: `A` : Concentration at depth `Z` [eggs/m^3]

If `Z` is present, `size(A) = size(Z)`, otherwise, `size(A) = size(ZE)`.

Description: Computes the exact stationary solution of the convection diffusion equation with constant eddy diffusivity `K` and velocity `W`. If `Z` is present, returns array of pointwise values. If `Z` is not present, returns exact cell averages.

eggvel – Terminal egg velocity

Usage: `[W, Re] = eggvel(drho, d, mu)`

Input:

`drho`	: Buoyancy of egg	[$\mathrm{kgm^{-3}}$]
`d`	: Diameter of egg	[m]
`mu` (opt)	: Dynamic molecular viscosity	[$\mathrm{kgm^{-1}s^{-1}}$]

`drho`, `d` and `mu` can be matrices (of the same shape). `mu` can be also be a scalar or omitted. The sign of `drho` is positive if the egg is ascending, `drho` = Density of water - density of egg. With only two arguments, a default value 0.0016 is used for `mu`.

Output:

`W`	: Terminal velocity	[$\mathrm{ms^{-1}}$]
`Re` (opt)	: Reynolds number	

Description: Computes the terminal velocity of a small sphere in sea water by the formulas in Stokes' or Dallavalles formula.

eggvelst – Egg velocity from salinity and temperature

Usage: `[W, Re] = eggvelst(S, T, d, Se)`

Input:

`S`	: Salinity of the environment	[psu]
`T`	: Temperature	[°C]
`d`	: Egg diameter	[m]
`Se`	: Egg salinity	[psu]

All arguments can be arrays of the same shape. Alternatively `d` and/or `Se` may be scalars.

Output:

`W`	: Terminal velocity	[$\mathrm{ms^{-1}}$]
`Re` (opt)	: Reynolds number	

Description:

Computes the terminal egg velocity given the hydrography of the environment and the salinity `Se` where the egg is neutral buoyant.

fluxlim – Numerical integration of transport equation

Usage: `A = fluxlim(A0, K, W, nstep, dt, P, alpha)`

Input:

`A0`	: Start concentration	[$\mathrm{eggs/m^3}$]
`K`	: Eddy diffusivity	[$\mathrm{m^2s^{-1}}$]
`W`	: Terminal velocity	[$\mathrm{ms^{-1}}$]
`nstep`	: Number of integration steps	
`dt`	: Time step	[s]
`P` (opt)	: Spawning term	[$\mathrm{eggs/m^3/s}$]
`alpha` (opt)	: Loss coefficient	[1/s]

A0 lives at the egg-points, `size(A0) = (Ncell x 1)`. K and W live at the flux-points, `size = (Ncell+1 x1)`. If P and alpha are present, they also live at the egg-points. If P and alpha are missing, the source term is ignored.

Output: A : Result concentration [eggs/m^3]

A lives at egg-points in the same way as A0

Description:

Integrates the convection-diffusion equation by the flux-limited method. Starting with the concentration in A0 nstep integration steps are performed. The result is saved in A.

ftcs – Numerical integration of transport equation

Usage: `A = ftcs(A0, K, W, nstep, dt, P, alpha)`

Input:			
	A0	: Start concentration	[eggs/m^3]
	K	: Eddy diffusivity	[m^{2}s^{-1}]
	W	: Terminal velocity	[ms^{-1}]
	nstep	: Number of integration steps	
	dt	: Time step	[s]
	P (opt)	: Spawning term	[eggs/m^3/s]
	alpha (opt)	: Loss coefficient	[1/s]

A0 lives at the egg-points, `size(A0) = (Ncell x 1)`. K and W live at the flux-points, `size = (Ncell+1 x1)`. If P and alpha are present, they also live at the egg-points. If P and alpha are missing, the source term is ignored.

Output: A : Result concentration [eggs/m^3]

A lives at egg-points in the same way as A0

Description:

Integrates the convection-diffusion equation by the forward time central space (FTCS) method. Starting with the concentration in A0 nstep integration steps are performed. The result is saved in A.

lwendrof – Numerical integration of transport equation

Usage: `A = lwendrof(A0, K, W, nstep, dt, P, alpha)`

Input:			
	A0	: Start concentration	[eggs/m^3]
	K	: Eddy diffusivity	[m^{2}s^{-1}]
	W	: Terminal velocity	[ms^{-1}]
	nstep	: Number of integration steps	

`dt`	:	Time step	[s]
`P` (opt)	:	Spawning term	[eggs/m^3/s]
`alpha` (opt)	:	Loss coefficient	[1/s]

`A0` lives at the egg-points, `size(A0) = (Ncell x 1)`. `K` and `W` live at the flux-points, `size = (Ncell+1 x1)`. If `P` and `alpha` are present, they also live at the egg-points. If `P` and `alpha` are missing, the source term is ignored.

Output: `A` : Result concentration [eggs/m^3]

`A` lives at egg-points in the same way as `A0`

Description:

Integrates the convection-diffusion equation by the Lax-Wendroff method. Starting with the concentration in `A0` nstep integration steps are performed. The result is saved in `A`.

molvisc – Dynamical molecular viscosity of sea water

Usage: `mu = molvisc(S, T)`

Input: `S`	:	Salinity	[psu]
`T`	:	Temperature	[°C]

`S` and `T` may be arrays of the same shape.

Output: `mu` : Dynamic molecular viscosity [kgm^{-1}s^{-1}]

`mu` is an array of the same shape as `S` and `T`.

Description:

Computes the dynamic molecular viscosity by formula (1.47).

spawn – Make concentrated egg distribution

Usage: `A = spawn(M, Z)`

Input: `M`	:	Vertical integrated concentration	[eggs/m^2]
`Z`	:	Spawning depth	[m]

`M` and `Z` are scalars.

Output: `A` : Egg distribution [eggs/m^3]

`A` is a column vector, living at the egg points.

Description:

Returns a vertical egg distribution `A` with vertical integral `M`, concentrated as much as possible around depth = `Z`. If `ZE(Ncell)` $<$ `Z` $<$ `ZE(1)`, then `Z = ve_mean(A)`.

srcsact – Stationary solution, const. coeff., source term

Usage: `A = srcsact(K, W, P, alpha, Z)`

Input:

`K`	:	Eddy diffusivity	$[\mathrm{m^2 s^{-1}}]$
`W`	:	Terminal velocity	$[\mathrm{m s^{-1}}]$
`P`	:	Egg production	$[\mathrm{eggs/m^3/s}]$
`alpha`	:	Egg loss rate	[1/s]
`Z` (opt)	:	Vertical coordinate	[m]

`K`, `W`, `P` and `alpha` are scalars. `Z` can be an arbitrary array. If `Z` is ommitted, `ZE` is used as vertical coordinate.

Output: `A` : Concentration at depth `Z`. $[\mathrm{eggs/m^3}]$

If `Z` is present, `size(Y) = size(Z)`, otherwise, `size(Y) = size(ZE)`.

Description:

Computes the exact stationary solution of the convection diffusion equation with constant eddy diffusivity `K`, velocity `W`, egg production `P` and loss rate `alpha`. If `Z` is present, returns array of pointwise values. If `Z` is not present, returns exact cell averages.

sstate – Steady state solution

Usage: `A = sstate(M, K, W)`

Input:

`M`	:	Vertical integrated concentration	$[\mathrm{eggs/m^2}]$
`K`	:	Eddy diffusivity	$[\mathrm{m^2 s^{-1}}]$
`W`	:	Egg velocity	$[\mathrm{m s^{-1}}]$

`M` is scalar, `K` and `W` are column vectors of length `Ncell+1`, `K` and `W` live at the flux-points.

Output: `A` : Stationary solution $[\mathrm{eggs/m^3}]$

`A` lives at the egg points, `size = (Ncell x 1)`.

Description:

Computes the steady state solution of the convection- diffusion equation without source term, with eddy diffusivity `K` and egg velocity `W` variable in the water column. The solution is computed by viewing `K` and `W` as piecewise constant on the grid cells.

upstream – Numerical integration of transport equation

Usage: `A = upstream(A0, K, W, nstep, dt, P, alpha)`

Input:

`A0`	: Start concentration	[eggs/m^3]
`K`	: Eddy diffusivity	[m^{2}s^{-1}]
`W`	: Terminal velocity	[ms^{-1}]
`nstep`	: Number of integration steps	
`dt`	: Time step	[s]
`P` (opt)	: Spawning term	[eggs/m^3/s]
`alpha` (opt)	: Loss coefficient	[1/s]

`A0` lives at the egg-points, `size(A0) = (Ncell x 1)`. `K` and `W` live at the flux-points, `size = (Ncell+1 x1)`. If `P` and `alpha` are present, they also live at the egg-points. If `P` and `alpha` are missing, the source term is ignored.

Output:

`A`	: Result concentration	[eggs/m^3]

`A` lives at egg-points in the same way as `A0`

Description:

Integrates the convection-diffusion equation by the upstream method. Starting with the concentration `A0`, nstep integration steps are performed. The result is saved in `A`.

ve_drint – Integrate over a depth range

Usage: `int = ve_drint(A, z1, z2)`

Input:

`A`	: Egg distribution	[eggs/m^3]
`z1`	: First integration limit	[m]
`z2`	: Second integration limit	[m]

`A` must be a matrix where the columns are egg distributions. But a row-vector can also be accepted, `size(A) = (Ncell x n)` or `(1 x Ncell)`.

Output:

`int`	: Integral of A over depth range	[eggs/m^2]

If `A` is a matrix of size `(Ncell x m)`, `int` becomes a row-vector of length `m` containing the integrals of the columns of A.

Description:

Computes the vertical integral

$$\texttt{int} = \int_{\texttt{z1}}^{\texttt{z2}} a(z)dz$$

where $a(z)$ is the piecewice constant function $a(z) =$ `A(i)` for `ZF(i+1)` $< z <$ `ZF(i)`. If `A` is a matrix, the integral is computed for each column

ve_grid – Set up vertical grid

Usage: `ve_grid(H, dz0)`

Input:

`H`	: Depth of water column	[m]
`dz0`	: Grid size	[m]

Output: None

Description:

Sets up the vertical grid used in VertEgg, given the depth `H` of the water column and the grid size `dz0`. (Re)defines all global variables. The global variables are declared by `ve_init`.

ve_init – Initialize VertEgg

Usage: `ve_init`

Input: none

Output: none

Description:

Script for initializing VertEgg. Declares the global variables, so they become available in the workspace. The actual values are set by `ve_grid`.

ve_int – Vertical integral of egg distribution

Usage: `M = ve_int(A)`

Input:

`A`	: Egg distribution	[eggs/m^3]

`A` must be a matrix where the columns live on egg-points, but a row-vector at egg-points is also accepted, `size(A) = (Ncell x n)` or `(1 x Ncell)`

Output:

`M`	: The vertical integral	[eggs/m^2]

If `A` is a matrix of size `(Ncell x n)`, `M` is a row-vector of length `n`.

Description:

Computes the vertical integral of an egg distribution. `ve_int` is a vector function.

ve_mean – Mean depth of an egg distribution

Usage: `mu = ve_mean(A)`

Input:

`A`	: Egg distribution	[eggs/m^3]

`A` must be a matrix where the columns live on egg-points, but a row-vector at egg-points is also accepted, `size(A) = (Ncell x n)` or `(1 x Ncell)`

Output: `mu` : The center of gravity [m]

If `A` is a matrix of size `(Ncell x n)`, `mu` is a row-vector of length `n`.

Description:

Computes the mean (or center of gravity) of the egg egg distribution `A`, `ve_mean` is a vector function.

ve_rand – Make random egg distribution

Usage: `Y = ve_rand(M)`

Input: `M` : Vertical integrated concentration [eggs/m^2]

`M` is a scalar

Output: `Y` : Random egg distribution [eggs/m^3]

`Y` is vector of size `[Ncell 1]`

Description:

A uniform distribution is used to generate random values between 0 and 1. The values are scaled to make `ve_int(Y) = M`.

ve_rmsd – Root mean square deviation

Usage: `R = ve_rmsd(X, Y)`

Input: `X` : Egg concentration [eggs/m^3]
`Y` : Egg concentration [eggs/m^3]

`X` and `Y` may be matrixes of the same size with `Ncell` rows. `X` and/or `Y` may also be vectors of length `Ncell`.

Output: `R` : Root mean square deviation [eggs/m^3]

`R` is a row vector with length = `max(columns(X), columns(Y))`.

Description:

Computes the root mean square deviation `R` between the columns of `X` and `Y`. If one argument is a vector, it is compared to all columns of the other argument. If `X` and `Y` are matrices

$$\mathtt{R(j)} = \sqrt{\frac{1}{\mathtt{Ncell}} \sum_{\mathtt{i}} (\mathtt{X(i,j)} - \mathtt{Y(i,j)})^2}.$$

If `Y` is a vector

$$\mathtt{R(j)} = \sqrt{\frac{1}{\mathtt{Ncell}} \sum_{\mathtt{i}} (\mathtt{X(i,j)} - \mathtt{Y(i)})^2}.$$

ve_std – Standard deviation of egg distribution

Usage: `s = ve_std(A)`

Input: `A` : Egg distribution [eggs/m^3]

`A` must be a matrix where the columns live on egg-points, but a row-vector at egg-points is also accepted, size(A) = (Ncell x n) or (1 x Ncell)

Output: `s` : The standard deviation [m]

If `A` is a matrix of size (Ncell x n), s is a row-vector of length n.

Description:

Computes the standard deviation of an egg distribution `A`, `ve_std` is a vector function.

Appendix A

Installation

A.1 Availability of the software

In the spirit of free exchange of scientific ideas, the VertEgg toolbox is free software. The software is available by anonymous ftp from the cite `ftp.imr.no` in the directory `/pub/software/VertEgg`.

The software can be freely redistributed and modified, the only restriction is that it can not be included in commercial or shareware software. The author and the Institute of Marine Research can of course not give any guarantie that VertEgg behaves as it should.

If you download and in particular if you use VertEgg the author would like to be informed. Use `e-mail:bjorn@imr.no`. The VertEgg software and/or this report should be refered to when publishing results obtained by the use of VertEgg.

A.2 Matlab under Microsoft Windows

Matlab is a commercial product of The MathWorks, Inc., The official web-page is `http://www.mathworks.com/`. Matlab is available for PC, Macintosh and UNIX workstations. Vertegg has been developed and tested on version 4.2c.1 on PCs running Windows 3.11 and Windows 95.

The VertEgg toolbox consist of two directories, the toolbox and the example directory. The toolbox directory can be placed anywhere, but a natural place is `C:\MATLAB\TOOLBOX\LOCAL\VERTEGG`. Matlab must be told about the location of the toolbox. This is best done by adding the the directory to the `matlabpath` in the master startup file `C:\MATLAB\MATLABRC.M` or your own startup file. Alternatively, the directory can be added by the `path` command in an interactive session. The examples directory is not required, and can be placed anywhere.

A.3 Octave under UNIX

A short description of Octave is taken from the README-file,

> Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically.
>
> Octave is free software; you can redistribute it and/or modify it under the terms of the GNU General Public Licence as published by the Free Software Foundation.

Octave is developed by J.W. Eaton from the University of Wisconsin. The official webpage is `http://bevo.che.wisc.edu/octave.html`. The program is available by anonymous `ftp` from the official site, `bevo.che.wisc.edu` in the directory `/pub/octave`. Source code and binaries for many UNIX workstations are available. The current version (August 1995) is 1.1.1.

Octave uses gnuplot as its graphic library. The VertEgg toolbox will work without the graphics, but most of the examples will not. To do any serious work with Octave, gnuplot must be installed before Octave. Gnuplot is also a useful program by itself.

From the frequently asked questions (FAQ) about gnuplot,

> Gnuplot is a command-driven interactive function plotting program. It can be used to plot functions and data points in both two- and three- dimensional plots in many different formats, and will accommodate many of the needs of today's scientists for graphic data representation. Gnuplot is copyrighted, but freely distributable; you don't have to pay for it.

Gnuplot has many authors and is available on all platforms. The official www homepage for gnuplot is `http://www.cs.dartmouth.edu/gnuplot_info.html`, the program is available by anonymous `ftp` from `ftp.dartmouth.edu` in the directory `/pub/gnuplot` and many other cites. The current version (August 1995) is 3.5 but beta-releases of version 3.6 are updated frequently.

The natural place for the VertEgg toolbox directory is `/usr/local/lib/octave/site/m/vertegg` where it automatically will be found by Octave. The examples directory can be placed anywhere.

Appendix B

Mathematical digressions

B.1 Exact solution of transient problem with constant coefficients

If the eddy diffusivity K and the egg velocity w are constant and the source term Q is zero, the convection-diffusion equation (1.4) can be solved analytically.

The problem is,

$$\varphi_t + w\varphi_z - K\varphi_{zz} = 0 \tag{B.1}$$

with boundary conditions

$$w\varphi - K\varphi_z = 0, \quad z = -H, z = 0. \tag{B.2}$$

Using a standard technique, separation of variables, let $\varphi(z,t) = A(t)B(z)$. With $m = w/K$, this gives the following ordinary differential equation for B,

$$B'' - mB' + \lambda B = 0, \tag{B.3}$$

or in Sturm-Liouville form,

$$(\mathrm{e}^{-mz}B')' + \lambda \mathrm{e}^{-mz}B = 0. \tag{B.4}$$

The boundary conditions are separated,

$$B' = mB, \quad z = -H, z = 0. \tag{B.5}$$

In this form we have a self-adjoint, regular Sturm-Liouville system.

To such a system, there is an increasing sequence of real eigenvalues, $\lambda_0 < \lambda_1 < \ldots$ with associated eigenfunctions $B_n(z)$. These functions forms an orthogonal system with respect to the inner-product

$$\langle f, g \rangle = \int_{-H}^{0} f(z)g(z)\mathrm{e}^{-mz}dz. \tag{B.6}$$

In our case the only eigenvalue $\lambda \leq m^2/4$ is $\lambda_0 = 0$ corresponding to the stationary solution $\varphi(z,t) = \mathrm{e}^{mz}$. For $\lambda > m^2/4$ let $\alpha^2 = \lambda - m^2/4$. The general solution to equation (B.3) is then

$$B(z) = \mathrm{e}^{\frac{1}{2}mz}(b_1 \cos(\alpha z) + b_2 \sin(\alpha z)), \tag{B.7}$$

Imposing the boundary conditions gives

$$2\alpha b_2 = m b_1, \quad \text{and} \quad \alpha_n = \frac{n\pi}{H}, \quad n = 1, 2, \ldots. \tag{B.8}$$

The normalised eigenfunctions with respect to the inner product are

$$\lambda_0 = 0, \quad B_0 = \sqrt{\frac{m}{1 - \mathrm{e}^{-mH}}}\mathrm{e}^{mz} \tag{B.9}$$

and for $n \geq 1$

$$\lambda_n = \frac{m^2}{4} + \left(\frac{n\pi}{H}\right)^2, \quad B_n = (2H\lambda_n)^{-\frac{1}{2}}\mathrm{e}^{\frac{1}{2}mz}(2\alpha_n \cos(\alpha_n z) + m \sin(\alpha_n z)) \tag{B.10}$$

For $\lambda_0 = 0$ the time dependent part A_0 is constant. The time dependent part corresponding to $\lambda = \lambda_n$ with $n \geq 1$ is

$$A_n(t) = \mathrm{e}^{-\lambda_n K t} \tag{B.11}$$

This gives the general format for the solution of equation (B.1),

$$\varphi(z,t) = \sum_{n=0}^{\infty} a_n \mathrm{e}^{-\lambda_n K t} B_n(z) \tag{B.12}$$

The coefficients a_n are determined from the initial values $\varphi(z,0) = f(z)$ by

$$a_n = \langle f, B_n \rangle = \int_{-H}^{0} f(z) B_n(z) \mathrm{e}^{-mz} dz. \tag{B.13}$$

B.2 Linear conservative schemes

The linear finite difference schemes considered in section 2.1.1 are all in conservation form (2.7)

$$(\varphi_i^{n+1} - \varphi_i^n)\Delta z = (F_{i+1}^n - F_i^n)\Delta t \tag{B.14}$$

where the flux function have the forms (2.8)–(2.9)

$$F_i = \alpha_i \varphi_i + b_i \varphi_{i-1} = \frac{\Delta z}{\Delta t}(a_i \varphi_i + b_i \varphi_{i-1}) \tag{B.15}$$

where the coefficients are independent of the φ_i-s.

B.2.1 Conditions for consistence

A numerical scheme for a partial differential equation must be *consistent*, this means that the scheme converges to the equation as the time and space steps approach zero.

Here the first expression $F_i = \alpha_i\varphi_i + \beta_i\varphi_{i-1}$ is used. The φ_i-s are defined as cell averages in equation 2.3 For continuous functions $\alpha(z)$, $\beta(z)$ and $\varphi(z)$, $F_i = \tilde{F}(z_i)$ where

$$\tilde{F}(z) = \frac{1}{\Delta z}\left(\alpha(z)\int_{z-\Delta z}^{z}\varphi(\zeta)\,d\zeta + \beta(z)\int_{z}^{z+\Delta z}\varphi(\zeta)\,d\zeta\right). \tag{B.16}$$

The Taylor expansion of φ around z is

$$\varphi(\zeta) = \varphi(z) + \varphi_z(z)(\zeta - z) + \frac{1}{2}\varphi_{zz}(z)(\zeta - z)^2 + \cdots. \tag{B.17}$$

Using this $\tilde{F}$ can be expanded as the series

$$\tilde{F} = (\alpha + \beta)\varphi - \frac{1}{2}(\alpha - \beta)\varphi_z\Delta z + \frac{1}{6}(\alpha + \beta)\varphi_{zz}\Delta z^2 - + \cdots. \tag{B.18}$$

For consistence, $\tilde{F}$ must converge to the flux function F,

$$\tilde{F} \to F = w\varphi - K\varphi_z \quad \text{as } \Delta z, \Delta t \to 0. \tag{B.19}$$

This gives the following consistence criterion

$$\alpha + \beta \to w \qquad \text{as } \Delta t, \Delta z \to 0 \tag{B.20}$$

$$(\alpha - \beta)\Delta z \to 2K \quad \text{as } \Delta t, \Delta z \to 0 \tag{B.21}$$

The schemes considered in section 2.1.1 should be consistent by construction. This is easily verified as $w = \alpha + \beta$ in all three cases and

$$(\alpha - \beta)\Delta z = \begin{cases} 2K & \text{FTCS,} \\ 2K + w^2\Delta t & \text{Lax-Wendroff,} \\ 2K + |w|\Delta z & \text{Upstream.} \end{cases} \tag{B.22}$$

B.2.2 Conditions for positivity and stability

For this analysis the nondimensional form

$$F_i = \frac{\Delta z}{\Delta t}(a_i\varphi_i + b_i\varphi_{i-1}) \tag{B.23}$$

of the numeric flux function is used.

Substituting equation (B.23) and the boundary conditions $F_i = F_{N+1} = 0$ in (B.14) gives the following expession for the solution φ^+ at the next time step

$$\varphi_1^+ = (1 + b_2)\varphi_1 + a_2\varphi_2, \tag{B.24}$$
$$\varphi_i^+ = -b_i\varphi_{i-1} + (1 - a_i + b_{i+1})\varphi_i + a_{i+1}\varphi_{i+1}, \quad i = 2, \ldots, N-1, \tag{B.25}$$
$$\varphi_N^+ = -b_N\varphi_{N-1} + (1 - a_N)\varphi_N. \tag{B.26}$$

For the scheme to be positive we must have $\varphi_i^+ \geq 0$ for all nonnegative values of $\varphi_{i-1}, \varphi_i, \varphi_{i+1}$. In other words, the coefficients above must all be nonnegative

$$a_i \geq 0, \quad i = 2, \ldots, N, \tag{B.27}$$
$$b_i \leq 0, \quad i = 2, \ldots, N, \tag{B.28}$$
$$-b_2 \leq 1, , \tag{B.29}$$
$$a_i - b_{i+1} \leq 1, \quad i = 2, \ldots, N-1, \tag{B.30}$$
$$a_N \leq 1. \tag{B.31}$$

If the coefficients a and b are constant, a traditional Von Neumann stability analysis can be carried out. The amplification factor is

$$\lambda = 1 - (a - b)\cos^2\Theta + \mathrm{i}(a + b)\sin\Theta \tag{B.32}$$

for $\Theta = m\pi\Delta z$. The stability condition $|\lambda| \leq 1$ for all values of Θ then becomes

$$(a + b)^2 \leq (a - b) \leq 1. \tag{B.33}$$

In this case, the positivity conditions, eqs. (B.27–B.31) are reduced to

$$a \geq 0, \quad b \leq 0, \quad a - b \leq 1, \tag{B.34}$$

which is is stronger than the stability condition.

B.2.3 Positivity with loss term

Egg production works to enhance the positivity of the solution while loss of eggs may ruin an otherwise positive solution. For the analysis of this situation we use equation (2.7) with F_i given by the nondimensional form (2.9) and the source term Q_i given by eq. (2.40). Substitutitint the F and Q terms in (2.7) gives the following genrealisation of eq. (B.25)

$$\varphi_i^+ = P_i\frac{1 - \mathrm{e}^{-\alpha_i\Delta t}}{\alpha_i} - b_i\varphi_{i-1} + (\mathrm{e}^{-\alpha_i\Delta t} - a_i + b_{i+1})\varphi_i + a_{i+1}\varphi_{i+1}, \quad i = 2, \ldots, N-1. \tag{B.35}$$

If this should be positive, regardless of P_i the conditions (B.29–B.31) must be strengthened

$$-b_2 \leq \mathrm{e}^{-\alpha_i\Delta t}, \tag{B.36}$$
$$a_i - b_{i+1} \leq \mathrm{e}^{-\alpha_i\Delta t}, \quad i = 2, \ldots, N-1, \tag{B.37}$$
$$a_N \leq \mathrm{e}^{-\alpha_i\Delta t}. \tag{B.38}$$

Appendix C

Future work

Hopefully, as the toolbox is put to use, it will develop to better suit the need of scientists in the field. Below is an unsorted list of things that may be included.

- Better numerical schemes for the convection-diffusion equation.
- Implementation of Lagrangian (particle tracking) approach.
- Implement analytical transient solution of convection-diffusion equation with constant coefficients.
- More functions for analysis of observed egg distributions.
- Non-uniform vertical grid.
- Ekman solver.
- Turbulence model(s).
- Homepage for VertEgg on World Wide Web.

Bibliography

J.W. Eaton. *Octave*, 1.1 edition, 1995. User manual, available by anonymous ftp from bevo.che.wisc.edu in directory /pub/octave.

C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume I. Springer-Verlag, 2 edition, 1991.

T. Haug, E. Kjørsvik, and P. Solemdal. Vertical distribution of Atlantic halibut eggs. *Can. J. Fish. Aquat. Sci.*, 41:798–804, 1984.

R.J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag, Basel, 2 edition, 1992.

P.P. Morgan. *SEAWATER: A library of MATLAB Computational Routines for the Properties of Sea Water*. CSIRO Marine Laboratories, 1994. Report 222.

J.P. Riley and G. Skirrow. *Chemical Oceanography*, volume 2. Academic Press, 2 edition, 1975.

P.J. Roache. *Computational Fluid Dynamics*. Hermosa Publishers, Albuquerque, 1972.

K. Sigmon. *MATLAB Primer*. CRC Press, 4 edition, 1994.

G.G. Stokes. On the effect of the internal friction on the motion of pendulums. *Cambridge Trans.*, 9, 1851.

S. Sundby. A one-dimensional model for the vertical distribution of pelagic fish eggs in the mixed layer. *Deep Sea Research*, 30:645–661, 1983.

S. Sundby. Factors affecting the vertical distribution of eggs. *ICES mar. Sci. Symp.*, 192: 33–38, 1991.

H.U. Sverdrup, M.W. Johnson, and R.H. Fleming. *The Oceans*. Prentice-Hall, 1952.

UNESCO. Tenth rep. of the joint panel on oceanographic tables and standards. UNESCO Tech. Pap. in Marine Science No. 36, 1981.

C.B. Vreugdenhil. Linear central finite-difference methods. In *Numerical Methods for Advection – Diffusion Problems* Vreugdenhil and Koren (1993).

C.B. Vreugdenhil and B. Koren. *Numerical Methods for Advection – Diffusion Problems*. Vieweg, Braunschweig, 1993.

T. Westgård. Two models of the vertical distribution of pelagic fish eggs in the turbulent upper layer of the ocean. *Rapp. P.-v. Réun. Cons. int. Explor. Mer*, 191:195–200, 1989.

C.-S. Yih. *Fluid Mechanics*. West River Press, Ann Arbor, 1977.